REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.			
1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** 11/20/1997	**3. REPORT TYPE AND DATES COVERED** Final Report—7/96-11/97	
4. TITLE AND SUBTITLE ONR-USC Research on Alternative Process Architectures For Procurement and Acquisition			**5. FUNDING NUMBERS** N00014-94-1-0889
6. AUTHORS Walt Scacchi			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Center for Service Excellence, Marshall School of Business University of Southern California Los Angeles CA 90089-1421			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Office of Naval Research Ballston Centre Tower One 800 North Quincy Street Arlington, VA 22217-5660			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES			
12a. DISTRIBUTION/AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED			**12b. DISTRIBUTION CODE**

13. ABSTRACT *(Maximum 200 words)*

In this project, researchers at the USC ATRIUM Laboratory engaged in a collaborative research project with the Office of Naval Research. This effort was a continuation of a project conducted with the Procurement Directorate at the Naval Air Warfare Center, Weapons Division, China Lake, CA. In both investigations, research effort was directed at understanding and redesigning business processes associated with military procurement and acquisition. In the latter case, effort focused on understanding and redesigning ONR's acquisition processes for managing research grants and contracts. A methodology for involving ONR personnel in eliciting, analyzing, and redesigning its acquisition processes was developed and frequently executed. USC researchers also investigated and developed prototype computer-based information systems that support the remote modeling, analysis, visualization, and execution of complex multi-person business processes. This led to the conception and demonstration of a new class of information system technologies called, "process-driven intranets." Results from this research project have been documented and published on the World Wide Web, academic journals and conference proceedings, as well as presented to multiple audiences at ONR Headquarters in Arlington, VA.

14. SUBJECT TERMS Acquisition reform, intranets, knowledge-based systems, process architecture, process life cycle, process prototyping, process redesign, procurement			**15. NUMBER OF PAGES** 104
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Computer Generated STANDARD FORM 298 (Rev 2-89) Prescribed by ANSI Std 239-18 298-102

19971201 052

USC Research on Alternative Process Architectures

for Procurement and Acquisition

Final Report

November 1997

Walt Scacchi, Principal Investigator
Director ATRIUM Laboratory
Center for Service Excellence
and
Information and Operations Management Dept.
Marshall School of Business
University of Southern California
Los Angeles, CA 90089-1421 USA
213-740-4782, 213-740-8494 (fax)
Scacchi@gilligan.usc.edu
Danie Mann
Administrative Contact
Center for Service Excellence
Marshall School of Business
USC
213-740-0185, 213-740-8494 (fax)
Dmann@sba.usc.edu

This document represents the final report for research sponsored by the Office of Naval Research under grant, N00014-94-0889.

This report consists of seven component reports which constitute the research results and principal deliverables produced with the support of this grant. The components reports are identified below, together with links to their Web-based versions. These reports constitute the remainder of this Final Report, and they appear in the order listed here.

1. *NAWCWPNS Procurement Process Architecture, Final Report, (July 1996)* which was presented and delivered to the technical POC at NAWC Weapons Division, Procurement Office. A Web-based version of this report including links to about 50 additional pages, images, and reports can be accessed on the World-Wide Web via the URL, **http://www.usc.edu/dept/ATRIUM/Papers/procure.html**
2. *ONR-USC Collaborative Research on Alternative ONR Process Architectures, Progress Report, (16 October 1996)*, which was presented and delivered to the technical POC and others at the Acquisition Directorate, ONR Headquarters, in Arlington, CA. A Web-based version of this report including links to about additional pages, images, and reports can be accessed on the World-Wide Web via the URL, **http://www.usc.edu/dept/ATRIUM/ONR-brief-Sept96.html**
3. *ONR-Darpa Intranet Briefing, (11 December 1996)*, which was presented and delivered to the technical POC, invited representatives from Darpa, and others at the Acquisition Directorate, ONR Headquarters, in Arlington, CA. A Web-based version of this report including links to about additional pages, images, and reports can be accessed on the World-Wide Web via the URL, **http://www.usc.edu/dept/ATRIUM/ONR-Darpa-brief-Dec96.html**
4. *Process Life Cycle Engineering: A Knowlege-Based Approach and Environment,* (March 1997) appears in *Intelligent Systems for Accounting, Finance, and Management,* Vol.5 (March 1997).A Web-based version of this report including links to about additional pages, images, and reports can be accessed on the World-Wide Web via the URL, **http://www.usc.edu/dept/ATRIUM/Papers/Process_Life_Cycle.html**
5. *(Re)Engineering Research Grants Management: From Acquisition Reform to Knowledge Brokering at ONR, (May 1997),* which was presented at the NSF Workshop on R&D for Federal Information Services, Arlington, VA 17-19 May 1997. A Web-based version of this report including links to about additional pages, images, and reports can be accessed on the World-Wide Web via the URL, **http://www.usc.edu/dept/ATRIUM/NSF-FIS-Workshop.html**
6. *Supporting Distributed Configuration Management in Virtual Enterprises,* (May 1997), which was published in R. Conradi (ed.), *Software Configuration Management,* Lecture Notes in Computer Science, Springer-Verlag, Berling and New York, 1997.A Web-based version of this report can be accessed on the World-Wide Web via the URL, **http://www.usc.edu/dept/ATRIUM/Papers/DHT-SCM7.ps**
7. *Process-Driven Intranets: Life Cycle Support for Process Reengineering,* which has been published in *IEEE Internet Computing,* Vol.1(5):42--49, September-October 1997.A Web-based version of this report can be accessed on the World-Wide Web via the URL, **http://www.usc.edu/dept/ATRIUM/Papers/PDI.ps**

NAWCWPNS Procurement Process Architecture

Final Report

July 1996

Walt Scacchi, Mark Nissen, and John Noll
ATRIUM Laboratory,
Information and Operations Management Dept.,
School of Business Administration,
University of Southern California.
213-740-4782, 213-740-8494 (fax)
(Scacchi@gilligan.usc.edu).

Preparation of the report and the research that preceding it were supported under a grant from the Office of Naval Research, N00014-94-0889. No endorsements implied.

Overview

- **Background and Definitions**

A paper providing an overall description of our process engineering life cycle can be found at http://www.usc.edu/dept/ATRIUM/Papers/Business_Process_Modeling.ps.

Background and Definitions

Strategic Issues

- High performance work organization and empowerment
- Efficient and effective business (procurement) process redesign and management
  - Cycle time and cost reduction
  - Agile response
  - Measurable process quality and performance
  - On-demand performance support ("just-in-training")
- Exploit emerging national information infrastructure

(Internet/WWW) for EDI, Electronic Funds Transfer (EFT), or Electronic Commerce (EC).

Tactical Problems

- **How to best support people who work on dynamic individual, group/team, and organizational assignments?**
- **How to model and redesign *resources* (work products, processes, organization, and information infrastructure) using a common representation?**
- **How to inter-relate products to processes to organizations to information infrastructure?**
- **How to integrate heterogeneous product data, tools/information systems, data repositories?**
- **How to accomodate wide-area resource distribution and organizational autonomy?**

Solution:

Near-term

- **Apply process engineering techniques and tools to analyze and identify procurement process redesign opportunities.**
- **Identify and transfer techniques, experiences, and lessons learned.**

Longer-Term

- Transition to a knowledge-based distributed hypertext (*KB-WWW*) infrastructure that supports the modeling, inter-relationship, integration, and accomodation of resources and people within local/global virtual procurement and acquisition organizations.

Definitions:

- Our focus in targeted at the engineering of complex business processes or capabilities *across their life cycle.*
- A *capability* represents the products, processes, organization staffing, and information infrastructure, as well as their interrelationships, for a recurring business activity.
- The *web of relationships* among the objects and attributes of a product, process, organization, infrastructure, or total capability defines its *architecture.*

A paper providing an exemplary description of a software development capability architecture can be found at http://www.usc.edu/dept/ATRIUM/Papers/Process_Meta_Model.ps.

Return to *Overview*

Sources of Experiences Encountered

- Anderson Consulting
- AT&T Bell Laboratories
- CoGenTex
- EDS
- Hewlett-Packard
- HoloSoFx (formerly Active Management Inc.)

- McKesson Water Products Co.
- Naval Air Warfare Center (China Lake, CA)
- Northrop(-Grumann) B-2 Division
- SUN Microsystems Computer Corp
- Perceptronics

Return to *Overview*

The process/capability life cycle follows.

Meta-Modeling

- **Constructing and refining a process concept vocabulary and logic (an ontology) for representing families of capabilities, processes, and their instances in terms of object classes, attributes, relations, constraints, control flow, rules, and computational methods.**
- ***Experience*: key to achieving *process-level interoperability*.**

Image files that show user interface displays of (a) a meta-model class hierarchy we use, and (b) a meta-model schema for the "task-force" class can be viewed when selected.

A paper providing a more detailed description of meta-modeling can be found at http://www.usc.edu/dept/ATRIUM/Papers/Process_Meta_Model.ps.

Return to *Overview*

Definition and Modeling

- **Eliciting and capturing of informal process descriptions, and** their conversion into formal process models or process model instances.

- *Experience*: Most "as-is" processes are ill-defined and not well understood.
- *Experience*: Most process redesign efforts want to primarily focus on "to-be" alternatives, without baselining as-is processes.

Image files that show views of the NAWCWPNS Procurement Process Architecture *can be found at the following locations:*

- *Organization*: The generic organizational capability for *Procurement* differs considerably across a range of organizations; however, nearly every organization shares common aspects of procurement activities.

- *Product*: The procurement product decomposition reveals the source of substantial variety within the NAWCWPNS Procurement Capability.

- *Technology*: The technology decomposition reveals some of the information systems employed in the NAWCWPNS Procurement Process, including APADE (which is being superseded/upgraded), and for EDI, SPEDI. We also suggest examining the Internet-based electronic commerce capability of FAST at USC-ISI, which might be adopted for future use.

- *Process*: The process decomposition outlines some of the important process elements of procurement at NAWCWPNS, including BankCard (which involves outsourced procurement), SPEDI (which involves electronic data interchange between NAWCWPNS and its vendors), and Large Negotiated Procurement (LNP) (which contains both the request for proposal (RFP) and Justification and Approval (J&A) as lower-level subprocesses).

Return to *Overview*

J&A Process

Image files that show views of the NAWCWPNS Justification and Approval (J&A) Process can be found at the following locations:

- *Level 1*: The process includes five activities or task chains and one quality feedback loop.

- *Level 2 - Customer Assistance*: The Customer Assistance process includes three activities and one quality feedback loop.

- *Level 2 - Approvals*: The Approvals process includes five task chains, one branch, and five quality feedback loops; each of the five task chains contains a nested quality feedback loop.

Return to *Overview*

RFP Process

Image files that show views of the NAWCWPNS Request for Proposal (RFP) Process can be found at the following locations:

- *Level 1*: This process includes seven activities or task chains, one branch, and one quality feedback loop.

- *Level 2*: This RFP Synopsis process includes four activities or task chains and one quality feedback loop.

- *Level 2*: This RFP Draft process includes five activities or task chains, one branch, and one quality feedback loop.

Return to *Overview*

LNP Process

Image files that show views of the NAWCWPNS Large Negotiated Procurement (LNP) Process can be found at the following locations:

- *Level 1*: This process includes four activities or task chains.
- *Level 2*: This LNP PrePalt process includes two activities or task chains, including the ARP subprocess, within which the J&A can be found.
- *Level 2*: This LNP PreAward process includes four activities or task chains, including the RFP subprocess.

Return to *Overview*

Analysis

- Evaluating static and dynamic properties of a process model, including its consistency, completeness, internal correctness, traceability, as well as other semantic checks.

- *Experience*: Best source of high-value, short-term results and payoffs.
- *Experience*: Easy to produce management reports or presentation materials.

Image files that show user interface displays of (a) a sample of process model analysis checks, (b) process model analysis statistics, and (c) process model analysis view, can be viewed when selected.

Return to *Overview*

Simulation:

Knowledge-Based

- Symbolically enacting process models in order to determine the path and flow of intermediate state transitions in ways that can be made persistent, replayed, queried, dynamically analyzed, and reconfigured into multiple alternative scenarios.
- *Experience*: High-value technology is infrequently used.
- *Experience*: Can produce narrative summaries of simulation runs.

Image files that show user interface displays of (a) a process model simulation interaction, and a subsequent (b) a process model simulation interaction can be viewed when selected.

A paper describing the initial design and implementation of this simulation mechanism can be found in, P. Mi and W. Scacchi, "A Knowledge-Based Environment for Modeling and Simulating Software Engineering Processes," *IEEE Trans. Knowledge and Data Engineering*, Vol. 2(3), 283-294, (1990). Reprinted in *Nikkei Artificial Intelligence*, Vol. 20(1), 176-191, (1991, in Japanese).

Discrete-Event

- Computationally enacting a sample of process models as network flows with hueristic or statistical arrival rates and service times so as to determine the overall process performance envelope, throughput, systematic behavior, and resource bottlenecks.
- *Experience*: Although less flexible, easy to use to discover process optimizations.
- *Experience*: Visual interactions and presentations always impress.

Image files that show user interface displays of a discrete-event process model workflow simulation for the Baseline J&A Process can be displayed on this link.

Return to *Overview*

Visualization

- Providing users with graphic views of process models and instances that can be viewed, navigationally traversed, interactively editted, and animated to convey process statics and dynamics.
- *Experience*: Process visualizations enable intuitive analysis and discovery.
- *Experience*: Visualization appears key to acceptability.

Image files that show user interface displays of (a) visual process model editor, (b) spreadsheet-like process browser, (c) graphical process model browser, and (d) graphical process object browser can be viewed when selected.

Return to *Overview*

Prototyping, Walkthrough, and Performance Support (Training On Demand)

- Incrementally enacting partially specified process model instances in order to evaluate process presentation scenarios to end users, prior to performing tool and data integration.
- *Experience*: Process prototyping and walkthrough is effective enabler for eliciting user feedback.
- *Experience*: Can provide a basis for user empowerment in controlling design and improvement of local processes.
- *Experience*: Generation of performance support materials in response to process improvements or changes is well-received.

***Image files** that show user interface displays of (a) a **process prototype** or **walkthrough display**, and (b) a* **web** ***of automatically generated (in HTML format) process performance support materials** that document the J&A capability can be browsed when selected.*

Return to *Overview*

Process Measurement and Diagnosis

- Process measurement is central to process redesign.
- Process measures can be qualitative, quantative, or calculated (e.g., as ratios of one measure to another, or as computed derivatives).
- A report on procurement process measurement can be reviewed for an introduction.
- *Experience:* Higher-order process quality measures such as *customer satisfaction* or *quality improvement* are best determined using computed derivatives.

- The following tables pertain to baseline and calculated measures for the classic J&A Process.

J&A Process Measurement Static Structure	
Measure	*Value*
Steps	31
Length	31
Breadth	1
Depth	4
Atomics	11
Branches	1
Iterations	11

J&A Process Measurement Static Organization	
Measure	*Value*
Organizational Roles	8
Value Chains	3
Department Handoffs	6
IT-supported activities	1
IT-supported comm.	0

J&A Process Measurement Static Derived		
Measure	*Value*	*Diagnostic*
Footprint	44	Small Process
Parallelism	1.00*	Linear Flow
Feedback Ratio	0.35	Complexity
Handoff Ratio	0.19	Departmental
IT-A Ratio	0.03	Manual Work
IT-C Ratio	0.00*	Paper Communication

Return to *Overview*

Process Redesign and Evaluation

- *Process measurements* and *domain-independent (re)design heuristics* can be used to envision a number of redesign alternatives.
- A *simulation interface screen* depicting one redesign alternative for the J&A Process - R1: Joint Review Meetings, can be seen here.
- The development and application of domain-dependent and instance-specific measures and heuristics is expected to produce *more specific redesign alternatives*, such as those shown here or below (with simulation results that forecast process cycletime reductions).

Process Redesign Cycletime Reductions: Simulation Forecast	
Redesign Alternative	*Time Reduction*
J&A Joint Reviews	67%
J&A Workflow System	67%
RFP Joint Reviews	22%
RFP Workflow System	28%
RFP Expert Review System	33%
RFP Composition KBS	67%
LCP Workflow System	21%
LCP Composition KBS	55%
LCP EDI-FAST	21-59%

- *Experience:* Redesign hueristics often produce differing improvements when applied to different processes. This means that we should not expect a single hueristic to equally improve all processes to which it is applied.

A paper describing our approach to process measurement as the basis for guiding process redesign can be found at http://www.usc.edu/dept/ATRIUM/Papers/Process_Measurement.ps.

Return to *Overview*

Next Steps and Future Directions

Return to *Overview*

Integration: Data, Tool, User Interface

- Encapsulating or wrapping new/legacy procurement information systems, repositories, and data objects that are to be invoked or manipulated when performing a procurement instance.
- A conceptual approach and system architecture that displays this can be found in a short presentation here.
- *Experience*: Can entail a lot of difficult technical work, but its relatively easy to construct "concept demo's."
- *Experience*: Growing interest in providing support for integration of wide-area heterogeneous information repositories using the Internet.
- Growing number of vendors are offering services and COTS

products that enable the integration of in-house information systems and databases with WWW-based browsers (as user interface to corporate intranet)

Image files that show user interface displays of (a) a to-do list, status, and tool associated with process actions that are integrated for use over the Internet.

A paper describing the mechanisms support data repository integration and a conceptual approach to procurement process enactment over the Internet can be found at http://www.usc.edu/dept/ATRIUM/Papers/VE_Procurement.ps.

A paper describing the strategy and mechanisms supporting process integration can be found in the paper, P. Mi and W. Scacchi, "Process Integration for CASE Environments," *IEEE Software*, Vol. 9(2), 45-53 (1992). Reprinted in *Computer-Aided Software Engineering*, 2nd. Edition, E. Chikofsky (ed.), IEEE Computer Society, (1993).

Return to *Overview*

Target Support Environment Generation

- Automatically transforming a process model or instance into a process-based computing environment that selectively presents prototyped or integrated information systems to end-users for process enactment.
- *Experience*: Considered a unique capability, not available in other process environments.
- *Experience*: Simplifies or eliminates low-level process programming via "application generator" techniques.

Image files that show user interface displays of (a) process-encapsulated tool environment that was generated via automated transformation of the modeled and integrated process.

A description of this mechanism can be found in the paper P.K. Garg, P. Mi, T. Pham, W. Scacchi, and G. Thunquest, "The SMART Approach to Software Process Engineering," *Proc. 16th. Intern. Conf. Software Engineering,*, IEEE Computer Society, Sorrento, Italy, 341-350, (1994).

Return to *Overview*

Instantiation and Enactment

- Performing the modeled process using the environment by a process engine that guides or enforces specified users or user roles to enact the process as planned.

Return to *Overview*

Enactment History Capture and Replay

- Graphically simulating the re-enactment of a process, in order to more readily observe process state transitions or to intuitively detect possible process enactment anomalies.
- *Experience*: Visualizing and replaying process enactment histories is well-received by managers and executives.
- *Experience*: Supports "organizational drill-down" when process anomalies are observed.

An image file that shows a user interface display of (a) process enactment event history and timing measurements can be viewed when selected.

Return to *Overview*

Conclusions

- Business process engineering in general, and procurement process engineering in particular, is a dynamic team-based

endeavor that can lead to mature, streamlined processes through:
  - understanding existing processes,
  - measuring and diagnozing process structure and throughput,
  - simulating or prototyping of process redesign alternatives,
  - incremental development, iterative refinement, and the reengineering of ad hoc process task instances and models.
- Process/capability engineering may be most likely to succeed when focused on *high frequency*, *short duration cycle time*, or *high velocity* processes.

Return to *Top*

ONR-USC Collaborative Research on Alternative ONR Process Architectures

Progress Report

***** Updated 16 Oct 1996 *****

Walt Scacchi
ATRIUM Laboratory
IOM Department
University of Southern California
Los Angeles, CA 90089-1421 USA
scacchi@gilligan.usc.edu
http://www.usc.edu/dept/ATRIUM

September 1996

This presentation can be found on the WWW at the URL:
http://www.usc.edu/dept/ATRIUM/ONR-brief-Sept96.html

Overview

- ONR Grants Management (GM) Process
  - Process capture and documentation
  - Process analysis

- Process redesign
- Wide-area process enactment environment (demo)
- Project research risk areas
- Future directions and options

Process capture and documentation

- Acquire qualitative process information (who, what, when, how, how much, how long, etc.)
  - *AS-IS* versions of
    - Pre-Award: Postscript diagram, HTML listing, or Process dictionary format.
    - Grant Award: Postscript diagram, HTML listing, or Process dictionary format.
    - Grant Administration: Postscript diagram, HTML listing, or Process dictionary format.
    - CloseOut: Postscript diagram, HTML listing, or Process dictionary format.

  captured and viewable as process decomposition diagrams (Postscript), HTML for browsing, or in PML (process dictionary definitions -- recommended for technical specialists only)
- Elicit understanding of current processes
  - Distributed across multiple, inconsistent documents
  - Refined from interviews with key personnel
  - Further refinement and validation by key process experts

- Educate process owners in our methodology, and how to document processes in computer-readable form
  - Future activity: Choose next set of ONR processes for redesign

---

Process analysis

- Examine organizational landscape and identify relevant issues and challenges
- Use knowledge-based tools/techniques for analysis
- Collectively select strategy for GM process redesign
  - ONR participation was essential and successful
  - Critical to identifying missing or overlooked GM steps (e.g., Make_Grant_Folder is first step of AS-IS Validate_PR_Package).
  - Opportunities for process streamlining have been identified by ONR participants.

Process redesign

- Articulate redesign approach
  - Model, analyze, diagnose (measure), identify relevant redesign hueristics, prioritize redesign alternatives,

prototype and demonstrate, transition

- Identify high leverage opportunity areas, expected outcomes, potential problems
  - Follows below
- Constrain redesign alternatives according to:
  - Do not increase any person's workload
  - Do not create new staff positions
  - Seek changes that are "self-motivating" or that enable organizational incentives.
- Model, visualize, and prototype new process alternatives

Process diagnosis and redesign alternatives for Grant Award

Diagnosis	*Applicable Hueristics*	*Expected ROI*
Manual step sequence	Consolidate and automate	Med-High
Linear step sequences	Identify parallelization opportunities	High
Many reviews steps	Joint reviews	High
Many data validation steps	Rule-based review system	Med-High
Many data validation steps	Push validation responsiblities upstream	Med-High
Manual assembly of compound documents	Rule-based document builder	Low-Med
Duplicating and circulating documents	Automate distribution and archiving	Med-Very High
Replace paper documents	Employ electronic proposals and grant documents	High-Very High
Islands of automation	Intranet with process support, data integration, and product navigation	Low-High
Wide-area workflow	Internet-based process enactment	Med-High

ONR Grants Management Process Transitions

Assign Priorities to Transitions

(1 = 30 days, 2 = 60 days, 3 = 90 days, 4 > 90 days)

Consult AS-IS process decomposition diagrams for details

Overall Suggestions for Pre-Award and Grant Award

- Consult with O3 to (a) identify and capture any Pre-Award improvements they can provide. For example, can they characterize types or groups of pre-approved, never-approved, insufficient-funds proposals? Automated award notice distribution? Input from O8 for not-approved proposals? Clarify responsibilities for who will verify or validate proposal funding and budget information?
- Develop strategy and plan for realizing "full" TO-BE vision. Initial version due 24 September 96.
- Need resolution and convergence of *electronic proposal* definition, design, implementation, and integration.

Pre-Award

- *Evaluate_Proposal* (1a): Update ONR Proposal Submission Guidelines on ONR WWW pages to solicit more complete proposal information (e.g., Identify BAA #, Certifications, Submittals). Consult Code 254 (Terry Young), ONR Webmaster (Code 92).
- *Evaluate_Proposal* (1b): Send Letter to Offeror (University

Business Office) requesting additional or missing proposal information; Mail returned to designated GS.

- *Approve_PR* (1c): Determine when O3 can use daily releases or IF actions. (Consult O3)
- *Approve_PR* (3): Automate PR Assignment to GS. Assignments to occur before proposal arrival. Deletes later (AS-IS) process step for manual Assigning PR to GS. Consult Code 21, 24, 25, 92.
- *Approve_PR* (2): Automate "Proposal_Received" Notice in Code 25. Consult Tricia.
- *Assemble_PR_Package* (1d): Eliminate Hard Copy PRs. This *deletes* AS-IS steps for Print, Sort, Compare, and Carry Hard Copy Proposals.
- *O2 Handle_Approved_Proposal* (3): Delete manual Assignment of PR to GS. Activity done earlier with automated support.
- *Validate_PR_Package* (1e): Delete Validate Hard Copy with PR. Discuss and clarify validation and verification responsibilities with O3 (already noted). Rename Verify_CAGE_SCCC_Codes to Verify/Obtain_Codes.
- *Obtain_Missing_Information* (1e): Delete FAX_Request Missing Information, unless 1a and 1b fail. Also, rename to Update_Missing_Data_Received.

Grant_Award

- *Print_Sign_Document* (2): Add Obligate Grant Data in STARS.
- *Prepare_Deliver_Package_to_FM* (2): Delete AS-IS steps for: Copy Cover Page, Copy FAD, Copy Signatures, Place on Pending Review Shelf, Log Data Sent to FM in INRIS, and

Deliever Package to FM. Consult Codes 08, 21, 25, 92.

- *Prepare_Deliver_Package_to_FM* (2): Delete Obligate Data in STARS (moved up earlier in process).
- *Distribution* (2): Delete Pickup Document, Log Date Returned, Match File.
- *Distribution* (4): Automate 100% Distribution (by 31 March 97). This depends on availability of adequate ADP information infrastructure. Consult Codes 08, 24, 25, 92, DFAS, and selected Universities for pilot tests.
- *Distribution* (4): Need approval to forego File Hard Copy Documents. Consult Codes 08, 25, and OOCC.

Overall suggestions for improvements in *Grant Adminstration* and *Close-Out* Processes

- All improvements to Grant Administration and Close-Out are priority "4" items, requiring more than 90 days to complete.
- ONR Field Operations perform Grant/Contract Administration and Grant/Contract Close-Out for ONR and other government agencies (e.g., Darpa, NASA), However, we only address improvements in ONR Grant Administration and Close-Out at this time. Handling Contracts and non-ONR grants is likely to benefit from the same kinds of improvements identified here, but getting them implementing is a major challenge.
- Improvements in Grant Administration Functions were not specified at this time, since these activities are demand-driven. Capturing these processes requires further

effort. Nonetheless, the provision of an electronic proposal and grant document folder was expected to help streamline the efficiency and effectiveness of these ongoing grant administration functions.

- Three transitions or transition events are needed to redesign nearly all of Grant Administration and Close-Out. These are
  - 4a: *Provision of Electronic Grants Packages using INRIS*. Scheduled for delivery by February 1997. This eliminates all tasks and actions involved in Grant Administration that receive, sort, distribute, and resolve problems with grants, except for the remaining need to handle new contractors, as they occur.
  - 4b: *Integration of CAMIS and INRIS*. Scheduled for completion by March 1997. This streamlines away all tasks and actions that involve the manual transfer and re-entry of information between INRIS and CAMIS.
  - 4c: *Provision of an Electronic Proposal and Grants Documents Folder*. Not yet scheduled. This represents a new technical capability that could be prototyped using USC's Distributed Hypertext (DHT) information infrastructure that could integrate INRIS, CAMIS, and other information systems together over a *Grants Management Intranet* for ONR.
- Develop strategy and plan for realizing "full" TO-BE vision.
  - Need resolution and convergence of *electronic proposal and grant folder* definition, design, implementation, and integration.

Proposed redesign of ONR GM Processes

- *TO-BE* version of ONR GM processes for
  - Pre-Award: Postscript diagram, HTML listing, or Process dictionary format.
  - Grant Award: Postscript diagram, HTML listing, or Process dictionary format.
  - Grant-Admin: Postscript diagram, HTML listing, or Process dictionary format.
  - Close-Out: Postscript diagram, HTML listing, or Process dictionary format.

  designed and presented as process decomposition diagrams.
- Note, in Grant-Award, if legal requirement for archiving Hard-Copy documents can be waived or removed, then Grant-Award process collapses to *three* steps (in contrast to 24 steps in AS-IS version)! Similarly, in Grant Administration, adoption of improvements with INRIS and its integration with CAMIS, Administration Preparation is streamlined from 21 steps in the AS-IS to 1 step in the TO-BE version!

Dynamically process modeling and prototyping

- Incrementally model and "enact" prototypes of process redesign alternatives
- Provide role-specific views of process task structure
- Demonstration of as-is version of Grant Award process (shown at USC)

- Develop prototype transition strategy and reimplementation (next).

Wide-area process enactment environment

- New process markup language (PML) for modeling process architectures
- PML compiler (generates process decomposition diagrams and WWW-based process enactment automatically)
- Internet-based execution system
- WWW-browser as Graphic User Interface (GUI) to process enactment environment
- Distributed hypertext-based object management system
- Integration mechanisms for incorporating new/legacy IT systems (INRIS, etc.)

Future directions and options

- Select, specify, and prototype GM redesign alternatives for Grants Administration and Grant Close-Out.
- Prototype and review demonstration GM process using candidate "electronic proposal" and "electronic proposal

document folder".

- Begin process capture and redesign of other ONR designated processes.
- Add automated support for process execution monitoring and audit trails
- Add (automated) support for process improvement process, remote analysis and simulation.
- Add on-demand process training mechanisms and content generation
- Integrate with OLE/ActiveX/ODBC, Java/JDBC, Oracle
- Extend PML to cover more complex conditions and responses
- Add support for automated process enactment guidance
- Add tools to facilitate capture and specification of processes in PML
- VRML-based GUI and process enactment
- Other training and transition support

ONR-Darpa Intranet Briefing

Walt Scacchi and John Noll
ATRIUM Laboratory
University of Southern California
Los Angeles, CA 90089-1421
scacchi@gilligan.usc.edu

11 December 1996

Overview

- Current situation
- Problems
- Goals
- Solution: a process-based intranet
- Proposed effort
- Downstream support issues
- Conclusions and next steps

Current Situation

- ONR, Darpa local-area networks operational; Internet access and WWW servers in use.
- Wide-area ONR "internet" operational; gateway connection to Internet, Email, etc.
- Therefore, *base* intranet components now in place,

- However, "islands of automation" (INRIS, CAMIS, STARS, etc.) now support ONR Grants Management processes.

Problems to address

- Difficult to efficiently manage and track Grants funding expenditures.
- Slow response to customer (e.g., Darpa POs) inquiries.
- Limited flexibility in expenditure management.
- Critical decision-making information not easily accessed.
- Limited external visibility of PR status or process performance.
- Heterogeneous computing platforms and legacy information systems limit timely information sharing.
- Intranet requires bridging of islands of automation with Internet technology.
- Limited support for policy and enforcement mechanisms.

Goals

- Establish internet foundation for ONR business processes, specifically the "Expenditure Management" process.
- Rapid response to customer (POs) inquiries.
- More agile and adaptive Expenditure Management.
- Timely access to critical PR status information for improved decision making.
- External visibility of performance on PR actions.
- Overcome platform and information system heterogeneity to

Key intranet *services*

- Email (including multi-media), News, and Bulletin Boards (BBoards).
- Corporate information directories and catalogs (Offerors, Investigators, POs, etc.).
- File management and printing
- Network management (access control, replication, client configuration, etc.).
- Audio (voice) and video messaging, conferencing, and groupware.
- Electronic commerce transactions (bank card, EDI/EFT, etc.).

What is a *process-based intranet*?

- *Adds* ability to (re)design, integrate, execute, and continuously improve business processes using an internet foundation.

A *demonstration* of a process-based intranet for ONR Grants Management

- This is a "concept demonstration".

facilitate access by remote decision makers.
- Establish foundation for growing and expanding network information-sharing systems and proceses.
- Establish policy and mechanisms for Expenditure data access, replication, and retention.

Solution

- Develop, demonstrate, refine, and deploy *process-based intranet* for ONR-Darpa PR Status and Expenditure Management.

What is an *Intranet*?

- *A TCP/IP network inside an organization linking its people, computers, applications, and information to improve information sharing.*

- Information sharing includes: access, creation and update, distribution, browsing, search and retrieval.

---

offeror, PI, etc.)
  - Darpa and other customers: PR status.
  - Determine current ONR internet performance factors.
  - Develop process-based intranet architecture plan.
  - Identify relevant policies for access control, replication, retention, etc.
  - Prototype and refine; implement and test; provide training and deploy process-based intranet.
- Develop cost estimate: Initial effort, $30,000 (300 person-hours effort @ $100/hour total burden) to develop ONR-Darpa PR Status and Expenditure Management process-based intranet.

Downstream support cost and investment items

- *Hardware platforms*: maintenance and replacement.
- *Network infrastructure*: providing increasing bandwidth; sourcing network services.
- *Intranet software*: maintenance ("point" solution), updates, and acquisition (e.g., Virtual Acquisition Library vs. in-house virtual file cabinet catalog).
- *Content*: maintenance and update distributed among departments and individuals.
- *Training*: Internet-based delivery and maintenance.
- New *support tasks* requirements.

Conclusions

- Can serve as a basis for developing a near-term capability (a "point" solution) supporting ONR-Darpa PR Status and Expenditure Management process employing the following suggested example reports.

	DARPA	PROCUREMENT	PROGRAM			
AWARD	ORDER	REQUEST	OFFICIER	ACQUISITION	OBLIGATED	AM(
NUMBER	NUMBER	NUMBER	CODE	SPECIALIST	FUNDING	INVC
N000149510986	D075	96PR03062-00	342	GALLON	100,000.00	100,(
N000149610680	D105	96PR05084-00	311	COLEMAN	160,000.00	100,(
N000149610854	C984	96PR05307-00	311	HILTZ	300,000.00	250,(
N000194920011	D128	96PR04842-02	361	FREEMAN	606,317.00	500,(
N000149520014	D050	96PR05343-00	36	WILLIAMS	381,562.00	81,5

				DATE		
			VENDOR	PROGRAM		DAT
DARPA	PROGRAM	PROCUREMENT	TECHNICAL	COUNCIL	DATE	ASSIGI
ORDER	OFFICIER	REQUEST	PROPOSAL	APPROVED	FORWARDED	TO
NUMBER	CODE	NUMBER	NUMBER	PR	TO ONR O2	SPECIA
B948	332	96PR02340-00	NONE	15-Nov-95	15-Nov-95	22-Nov
B958	312	96PR05092-00	NONE	28-Feb-96	28-Feb-96	29-Feb
B967	332	P6PR05946-00	NONE	3-Apr-96	3-Apr-96	9-Apr
C192	332	96PR04910-00	NONE	21-Feb-96	21-Feb-96	14-Mai
C209	313	96PR07868-00	NONE	8-Aug-96	14-Aug-96	16-Aug

Proposed effort

- Identify requirements for ONR intranet: Initial effort completed November 1996.
  - o Internal ONR Codes: expenditure rates (program,

We identified and characterized:

- Current situation and problems
- Attainable goals
- Proposed solution
- Demonstration of concept demonstration
- Proposed effort and cost
- Downstream support issues

Next Steps

- Decision and commitment of resources to persue proposed effort and solution.
- Decision on effort provider(s).
- Evaluate ONR internet performance.
- Planning for downstream support.

Process Life Cycle Engineering: A Knowledge-Based Approach and Environment

Walt Scacchi and Peiwei Mi
Information and Operations Management Department
University of Southern California
Los Angeles, CA 90089-1421
scacchi@gilligan.usc.edu
Voice: 213-740-4782, Fax: 213-740-8494

Abstract:

We describe our approach and mechanisms to support the engineering of organizational processes throughout their life cycle. We describe our current understanding of what activities are included in the process life cycle. We then go on to describe our approach, computational mechanisms, and experiences in supporting many of these life cycle activities, as well as compare it to other related efforts. Along the way, we present examples drawn from a recent study that uses the approach and the mechanisms of our knowledge-based process engineering environment to support the (re)engineering of corporate financial operations in a mid-size comsumer products organization.

Keywords: Business process reengineering, knowledge-based process engineering, process life cycle, knowledge-based process engineering environment

---

Revised Version, March 1997
To appear in *Intern. J. Intelligent Systems for Accounting, Finance, and Management*, 1997.

Introduction

Workflow modeling, business process redesign, enterprise integration, teamwork support, and management of knowledge assets are among the current generic goals for advanced information technology (IT) within organizations. Organizations are looking for ways to respond to competitive pressures and new performance levels by redesigning and continuously improving their production and operational processes. Organizations are also looking into IT as a strategy for establishing, sustaining and expanding presence in electronic markets for their goods and services. Such endeavors must therefore address complex organizational processes that entail tens, hundreds, or even thousands of organizational participants, as well as support the integration of a heterogeneous collections of both legacy and emerging ITs. Thus, we are faced with the problem of how to realize these goals in a coherent, scalable, and evolutionary manner.

In this paper, we describe the approach and supporting mechanisms we have been investigating at the USC ATRIUM Laboratory in an effort to solve this problem and realize these goals. As such, we describe our approach to the engineering and redesign of complex organizational processes using a knowledge-based computing environment, as well as some of the associated technologies we haved developed and deployed in large-scale business and government organizations. In particular, we draw upon examples resulting from the application of our approach and supporting knowledge-based environment to business processes found in corporate financial operations in a mid-size consumer products company (annual revenue more than $250M/year).

The Process Engineering Life Cycle

In simplest terms, we see that support for organizational processes entails more than the modeling and creation of process descriptions or representations. Our view is that the goal should be to support the engineering of organizational processes across the *process life cycle*. Much like the way that the development of complex information systems entails more than programming, so does the development of complex organizational processes entail more than creating models which describe them. As such, our work at USC has led to the initial formulation of an organizational process life cycle that is founded on the incremental development, iterative refinement, and ongoing evolution of organizational process descriptions. In this way, the organizational process life cycle spiral includes activities that address the following set of activities:

- *meta-modeling*: constructing and refining a process concept vocabulary and logic for representing families of processes and process instances in terms of object classes, attributes, relations, constraints, control flow, rules, and computational methods.
- *modeling*: eliciting and capturing of informal process descriptions, then converting them into formal process models or process model instances.
- *analysis*: evaluating static and dynamic properties of a process model, including its consistency, completeness, internal correctness, traceability, as well as other semantic checks. Also addresses the feasibility assessment and optimization of alternative process models.
- *simulation*: symbolically enacting process models in order to determine the path and flow of intermediate state transitions in ways that can be made persistent, replayed, queried, dynamically analyzed, and reconfigured into multiple alternative scenarios.
- *redesign*: reorganizing and transforming the structure of relationships within a process to compress completion time, as well as reduce or eliminate the number of steps, handoffs, or participants.
- *visualization*: providing users with graphic views of process models and instances that can be viewed, navigationally traversed, interactively editted, and animated to convey an intuitive understanding of process statics and dynamics.
- *prototyping, walkthrough, and performance support*: incrementally enacting partially specified process model instances in order to evaluate process presentation scenarios through the involvement of end users, prior to performing tool and data integration.

- *administration*: assigning and scheduling specified users, tools, and development data objects to modeled user roles, product milestones, and development schedule.

- *integration*: encapsulating or wrapping selected information systems, repositories, and data objects that can be invoked or manipulated when enacting a process instance. This provides a computational workspace that binds user, organizational role, task, tools, input and output resources into "semantic units of work".

- *environment generation*: automatically transforming a process model or instance into a process-based computing environment that selectively presents prototyped or integrated information systems to end-users for process enactment.

- *instantiation and enactment*: performing the modeled process using the environment by a process instance interpreter that guides or enforces specified users or user roles to enact the process as planned.

- *monitoring, recording, and auditing*: collecting and measuring process enactment data needed to improve subsequent process enactment iterations, as well as documenting what process steps actually occurred in what order.

- *history capture and replay*: recording the enactment history and graphically simulating the re-enactment of a process, in order to more readily observe process state transitions or to intuitively detect possible process enactment anomalies or improvement opportunities.

- *articulation*: diagnosing, repairing, and rescheduling actual or simulated process enactments that have unexpectedly broken down due to some unmet process resource requirement, contention, availability, or other resource failure.

- *evolution*: incrementally and iteratively enhancing, restructuring, tuning, migrating, or reengineering process models and process life cycle activities to more effectively meet emerging user requirements, and to capitalize on opportunitistic benefits associated with new tools and techniques.

- *process asset management*: organizing and managing the collection of meta-models, models, and instances of processes, products, tools, documents, and organizational structures/roles for engineering, redesign, and reuse activities.

While such a list of activities might suggest that engineering a business process through its life cycle proceeds in a linear or *waterfall* manner, this is merely a consequence of its narrative presentation. In practical situations where we have employed these activities and associated process mechanisms (e.g., at AT&T Bell Laboratories [Vo93], Northrop-Grumman Corporation, Naval Air Warfare Center (China Lake, CA), McKesson Water Products Company, and elsewhere [SM93]), it quickly becomes clear that business process engineering is a dynamic team-based endeavor that can only lead to mature processes through rapid process prototyping, incremental development, iterative validation and refinement, and the reengineering of ad hoc process task instances and models. To no surprise, many of our efforts addressing these life cycle activities and supporting prototype mechanisms have been described in greater detail elsewhere [MS90,MS91,NS91,MS92,MS93,MS96,SM93].

Each of these activities is neccesary to address a clear and distinct problem that emerges when using a process engineering environment to support a redesign effort. For example, with an early version of our process engineering environment [MS90], we sought to understand complex processes via modeling, analysis, and simulation. This enabled us to construct knowledge-based models of multi-agent business processes, which we could then analyze through queries and simulation. However, trying to convey what was modeled, or to explain the simulation results, we found that others not involved in directly using the environment could often not readily grasped how we acheived our computational results. This in turn then gave rise for a need to improve the communicability of results through the development and use of tools to support model visualization and visual simulation animations. Similarly, as the computational results of process simulation studies became more accessible and more easily understood, we encountered requests to be able to let other users engage, try out, or "fly" process simulations as a way to more effectively provide feedback about what process redesign alternatives made most sense to them. This in turn gave rise to the development of computational mechanisms for process prototyping and process enactment activities [MS92], and later still for automatically generating process enactable application environments [GM94]. Accordingly, each of the process life cycle engineering activities that we have investigated could generally be traced back to feedback from corporate users or others in our audience. Thus, while our approach and experience have led to a complex set of process life cycle engineering activities and supporting environment capabilities, each was found to be useful and necessary to address some particular process engineering need. Nonetheless, our experience as we will describe also suggests that it may still be the unusual circumstance where all process life cycle engineering activities are persued with comparable effort. But this may also just reflect the newness of the innovative capabilities we have at hand.

To date, our most substantial results of near-term value to businesses has been in supporting "upstream" process engineering activities (meta-modeling through redesign), while much of our recent research attention has been directed at activities from redesign through evolution and asset management. As such, we now turn to briefly describe our approach to some of these activities, with particular emphasis directed to the upstream engineering of business processes common to many corporate financial operations. We follow with description of selected downstream process engineering activities, and then compare our approach to other related research efforts.

Upstream Process Engineering: Meta-Modeling, Modeling, Analysis, Simulation and Redesign

We have developed a knowledge-based computing environment for engineering complex organizational processes [MS90]. We call this environment the Articulator. It first became operational in 1988, and we have continued to use and evolve it since. The Articulator utilizes a rule-based object-oriented knowledge representation scheme for modeling interrelated classes of organizational resources [MS90,MS96]. In this sense, the Articulator's knowledge representation ontology represents a resource-based theory of organizational processes, which in turn is in line with one of the principal basis for strategic planning and business management (cf. [Gr91,Boa93]). The Articulator's object classes characterize the attributes, relations, and computational methods associated with a taxonomy of organizational resources. Thus, using the Articulator, we can construct or prototype knowledge-based models of organizational processes.

Figure 1 provides a graphic overview of common corporate financial operations that are situated between internal customers and external vendors. Note that many sub-processes, such as those for order fulfillment and accounts payable, are not shown. However, for our ontology to be useful in such a domain, it must provide the concepts and representational constructs that allow all resources indicated or implied. This includes multiple decomposable tasks that potentially involve multiple actors or agents working within or across organizational units on different types of documents (purchase orders, invoices, etc.) with information systems. Furthermore, we would like such an ontology to be domain-independent, and thus be useful with little or no modification is supporting diverse process domains such as large-scale software system development, insurance claims processing, military procurement, and others [Sc89].

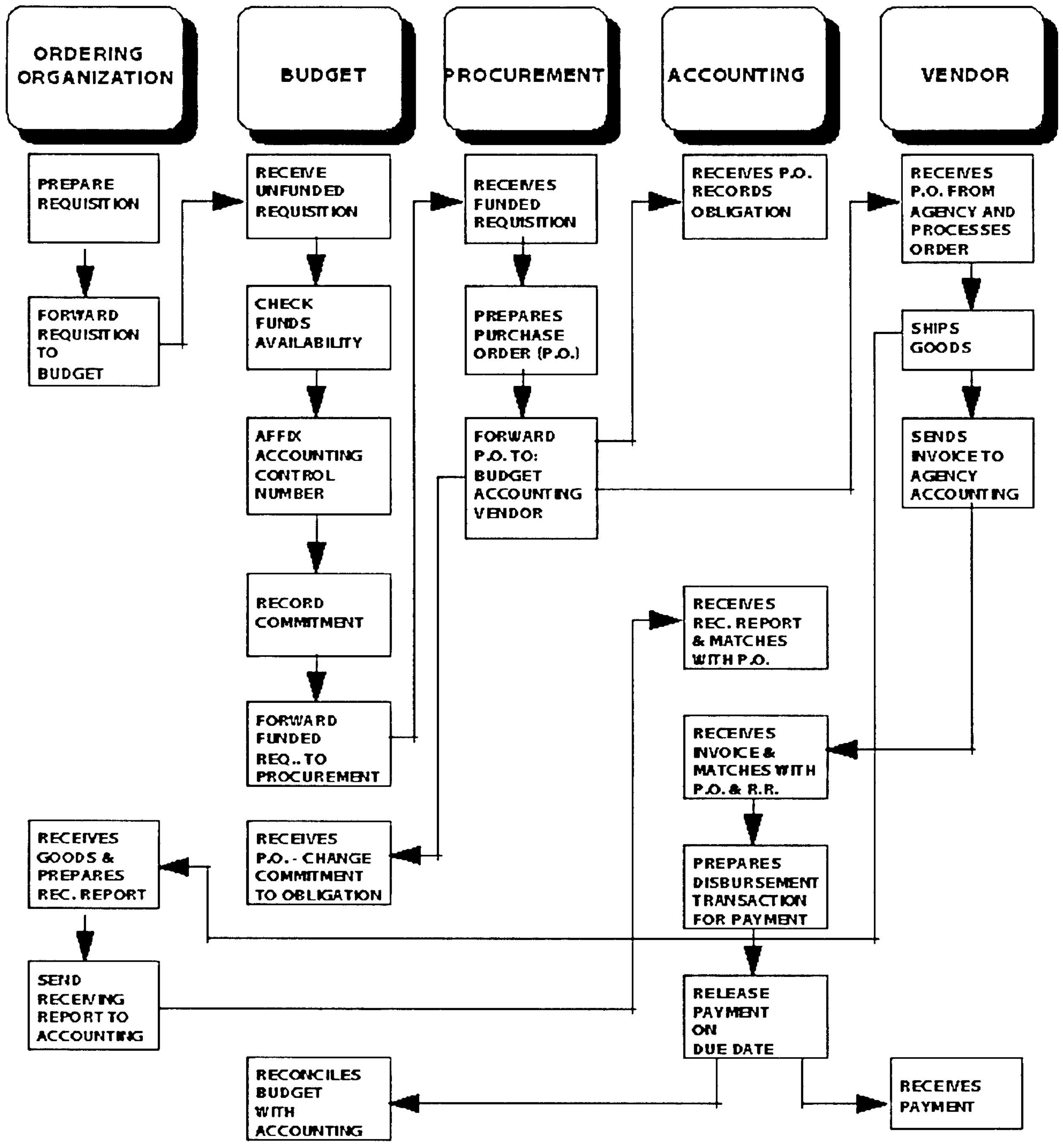

ORDERING ORGANIZATION
BUDGET
PROCUREMENT
ACCOUNTING
VENDOR
PREPARE REQUISITION
FORWARD REQUISITION TO BUDGET
RECEIVE UNFUNDED REQUISITION
CHECK FUNDS AVAILABILITY
AFFIX ACCOUNTING CONTROL NUMBER
RECORD COMMITMENT
FORWARD FUNDED REQ.. TO PROCUREMENT
RECEIVES FUNDED REQUISITION
PREPARES PURCHASE ORDER (P.O.)
FORWARD P.O. TO: BUDGET ACCOUNTING VENDOR
RECEIVES P.O. RECORDS OBLIGATION
RECEIVES P.O. FROM AGENCY AND PROCESSES ORDER
SHIPS GOODS
SENDS INVOICE TO AGENCY ACCOUNTING
RECEIVES REC. REPORT & MATCHES WITH P.O.
RECEIVES INVOICE & MATCHES WITH P.O. & R.R.
PREPARES DISBURSEMENT TRANSACTION FOR PAYMENT
RELEASE PAYMENT ON DUE DATE
RECEIVES PAYMENT
RECEIVES GOODS & PREPARES REC. REPORT
SEND RECEIVING REPORT TO ACCOUNTING
RECEIVES P.O. - CHANGE COMMITMENT TO OBLIGATION
RECONCILES BUDGET WITH ACCOUNTING

Figure 1:A view of Corporate Financial Operations (from [EC94])

Meta-Modeling and Modeling

The resource taxonomy we have constructed, explained in detail elsewhere [GS89,MS90,MS96], serves as a *process meta-model* which provides an ontological framework and vocabulary for constructing *organizational process models* (OPMs). Such an ontology is organized as a semantic network of object class schemata that define the name, attributes, relations and rule-based computational methods associated with each class of resource. For example, Figure 2 displays a schema definition for the TASK-FORCE object class, which is a resource sub-class used to define the structure of a business process, as well as to characterize its possible set of attributes and common relations.

```
{{ TASK-FORCE
      IS-A: SIMPLE-RESOURCE
      IS-A+INV: ACTIVITY TASK-CHAIN PRODUCTION-LATTICE
      CONTROL: NONSHARED
      STATUS: ALLOCATED
      ARTICULATING-STATUS: NON-AGENDAED
      TASK-FORCE-HAS-SCHEDULE:
      TASK-FORCE-CONTROLLED-BY-AGENT-ROLE:
      TASK-FORCE-COMPONENT-OF:
      TASK-FORCE-HAS-PREDECESSOR:
      TASK-FORCE-HAS-SUCCESSOR:
      TASK-FORCE-TOP-PERFORMED-BY-AGENT-ROLE:
      TASK-FORCE-ASSIGNED-TO-AGENT-ROLE:
      TASK-FORCE-REQUIRE-TOOL-RESOURCE:
      TASK-FORCE-REQUIRE-RESOURCE:
      TASK-FORCE-PROVIDE-RESOURCE:
      TASK-FORCE-USING-RESOURCE:
      TASK-FORCE-BEING-PERFORMED-BY-COLLECTIVE-AGENT:
      TASK-FORCE-AS-EXPERIENCE:
      TASK-FORCE-AS-SKILL:
      TASK-TYPE:
      EARLIEST-START-TIME: -1
      LATEST-START-TIME: -1
      SLACK:-1
      OPTIMISTIC-DURATION: 0
      MOST-LIKELY-DURATION: 0
      PESSIMISTIC-DURATION: 0
      AVERAGE-DURATION: 1
      REMAINING-DURATION: 0
      DUE-TIME: 0
      ACTUAL-START-TIME: 0
      ACTUAL-FINISH-TIME: 0
      TASK-FORCE-COLLECTIVELY-AGENDAED:
      ITERATION-NEST-LEVEL: 0
}}
```

Figure 2: A class schema for the TASK-FORCE resource used in defining processes.

In this schema, a task-force (process) among other things has relations that define whether it is scheduled; controlled by some agent; part of some embedding process (a sub-process); preceded or followed by other processes; managed by some organizational authority; assigned to some agent within some organizational collective; and involve the use or manipulation of tools, inputs, outputs, experiences and skills. Furthermore, as a schedulable object, then it also has properties that indicate constraints and defaults that can guide scheduling mechanisms. Finally, values that can fill these relations or attributes may be other resource classes, or class instances.

In simplest terms, the process meta-model states that organizational processes can be modeled in terms of (subclasses of) agents that perform processes using tools which consume or produce resources. Further, agents, tools, and tasks are resources, which means they can also be consumed or produced by other agents and tasks. Using this framework, we can then construct classes of OPMs for different business processes or process domains. For example, an OPM for a generic accounts payable (AP) process may model the following kinds of relations: the AP department manager may produce staff through staffing and allocation tasks (sub-processes) that use funds required from the departmental budget. In turn, these staff may then be assigned to other creative or routine production tasks (handling invoices) using the provided resources (e.g., computer workstations, corporate financial information systems, spreadsheet and desktop publishing packages, schedules, and salary) to contruct the desired products or services (e.g., reports and documents). *OPM Instances* can then be created by binding values of corresponding real-world entities to the classes of corresponding entities employed in the OPM. For instance, Mary may be the AP department manager who is responsible for producing a weekly report on unresolved invoices (payable accounts) as part of a briefing to the Head of Accounting and others in senior management, possibly including the Chief Financial Officier. Mary's administrative authority enables her to assign 2-3 individuals in her department to use their desktop PCs that run Windows95 to invoke AP functions on the corporate financial system, Lotus 1-2-3 for spreadsheet calculations, and Powerpoint software in order to get the reports and presentation materials produced by the end of the week.

In OPMs and their instances, the agents, tasks, product resources, tools, and systems are all hierarchically decomposed into subclasses that inherit the characteristics of their (multiple) parent classes for economy in representation. Further, these resource classes and subclasses are interrelated in order to express relationships such as precedence among tasks (which may be sequential, iterative, conditional, optional, or concurrent), task/product pre- and post-conditions, authority relationships among agent in different roles, product compositions, information system/tool aggregations, and others [MS90,MS96]. Figure 3 provides a partial view of the functional decomposition of an AP subsystem components within a financial system from the vendor, JDEdwards.

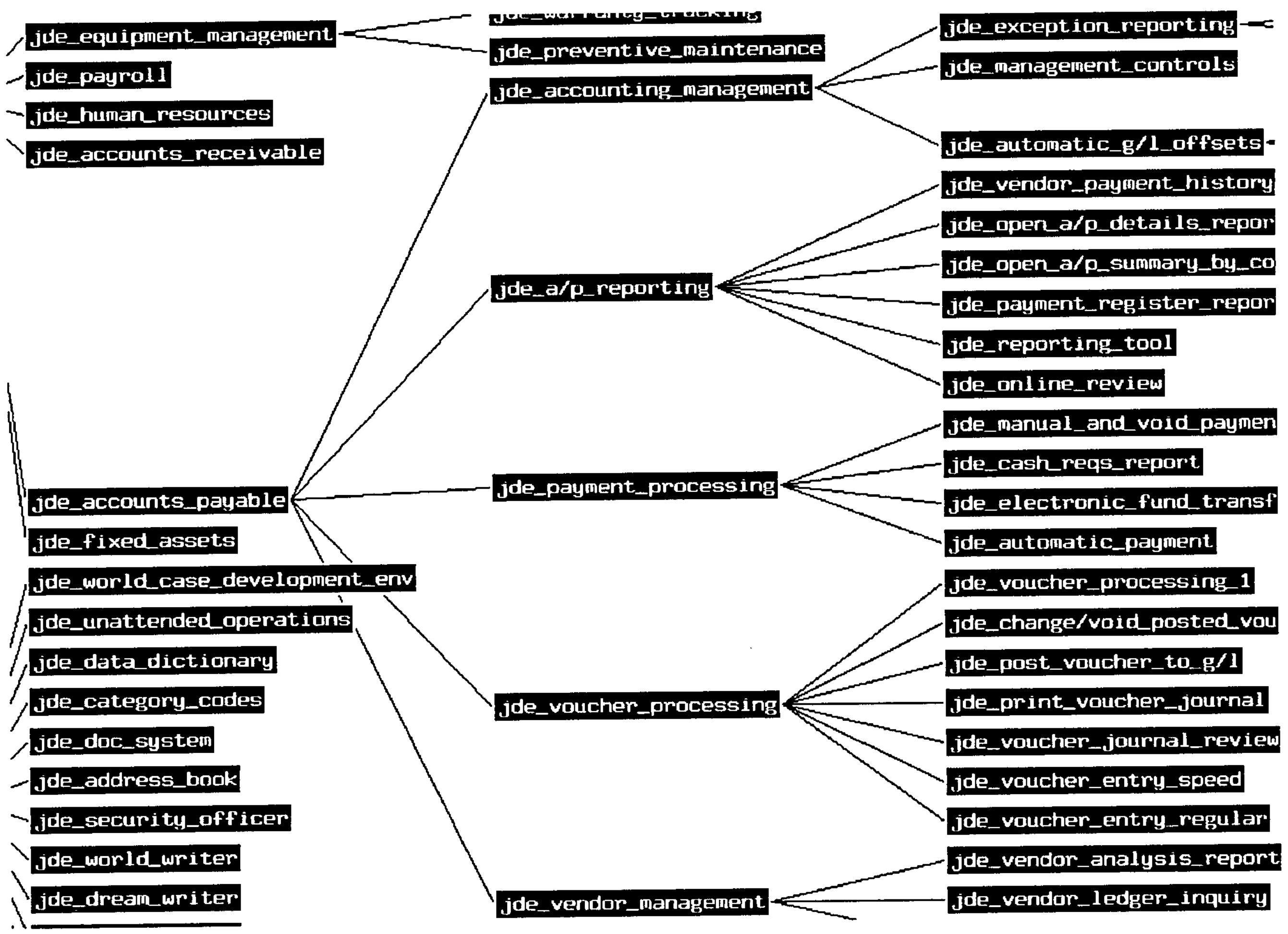


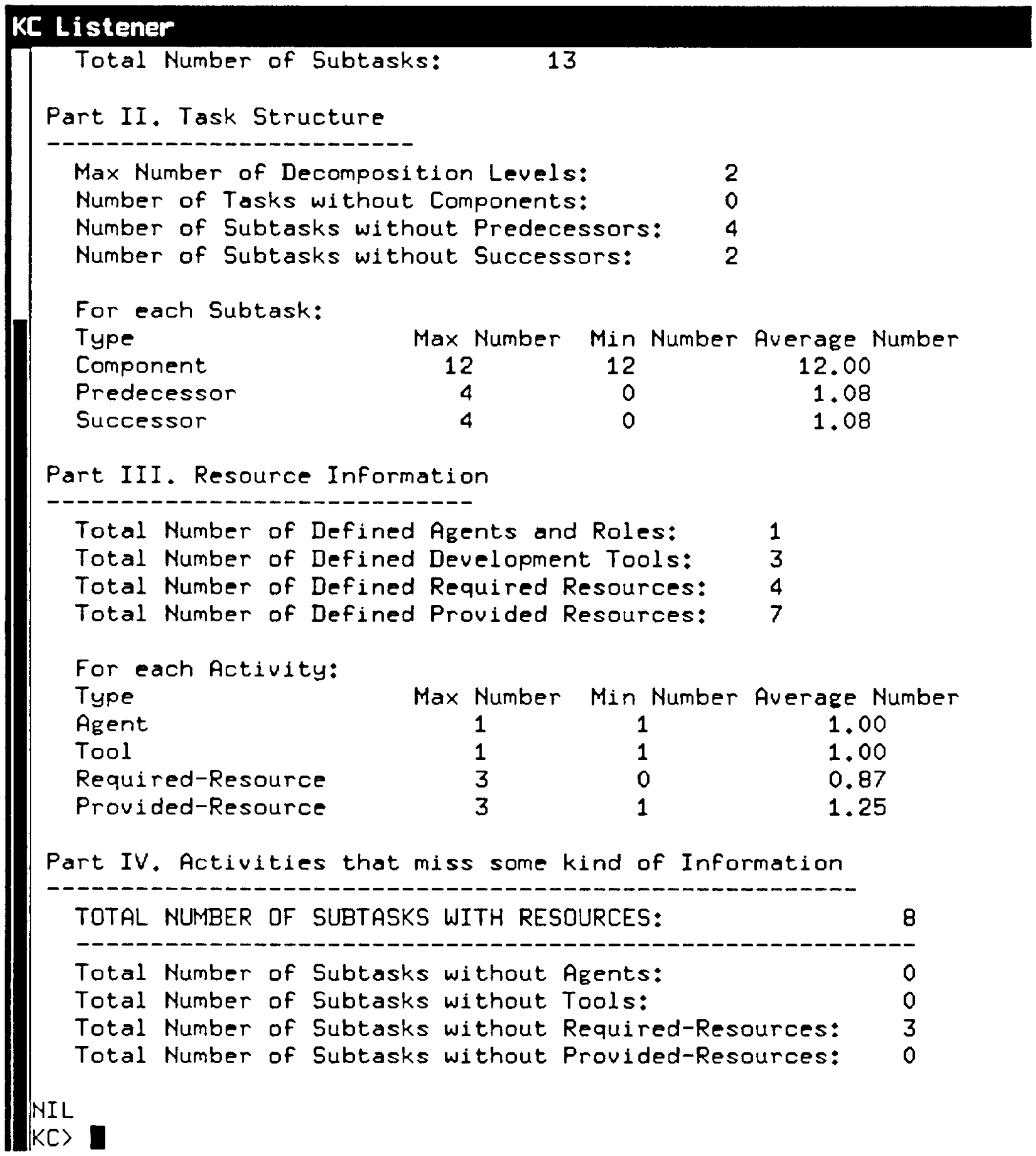
Figure 3: Hierarchical decomposition of AP financial system functional components

Accordingly, when using these classes of process modeling entities, we are naturally led to model organizational processes as a web of multiple interacting tasks that are collectively performed by a team of workers using an ensemble of tools that consume resources and produce composed products/artifacts [KS82]. In addition, it allows us to treat these models as *a reusable information resource* or *knowledge asset*, which can be archived, shared, generalized, or specialized for use in other organizations [LA94,SZ95].

Nonetheless, given the richness of the process meta-model and modeling representations, we must then face the challenge of iteratively eliciting, codifying, and revising actual OPMs and instance values from people who are experts in their business process domains. This is the knowledge acquisition bottleneck we must endure. We have found that it is usually necessary to conduct 2-4 rounds of interviews with people who are knowledgable about their processes. Furthermore, we choose to elicit knowledge about both existing "as-is" processes, as well as possible "to-be" process alternatives to help us better understand and represent the organizational knowledge at hand. Our experience has been that as-is business processes are ill-defined and not well understood, while most process experts or informants want to focus on to-be alternatives without baselining the current as-is conditions. While this experience may be common to developers of expert systems, the lack of as-is baseline can undercut the effort to systematically analyze and identify process redesign alternatives or transformation sequences, as well as to undercut the quantification of potential savings.

Analysis

As the process meta-model provides the semantics for OPMs, we can construct computational functions that systematically analyze the consistency, completeness, traceability and internal correctness of OPMs [MS90,CS96]. Such functions help in *verifying* logical and semantic properties of OPMs we capture, as well as revealing the presence of gaps or bugs in the emerging OPMs. These functions represent batched or interactive queries through the Articulator to the knowledge base through its representational schemata. We have defined a few dozen paramaterized query functions that can retrieve information through navigational browsing, direct retrieval, or deductive inference, as well as what-if simulations of partial or complete OPMs [MS90]. For example, in Figure 4, we show part of the results from a query function applied to an OPM that calculates some descriptive statistics and reports a tally of the number and types of incomplete resource specifications that were detected.



```
    Total Number of Subtasks:       13

  Part II. Task Structure
  -------------------------
    Max Number of Decomposition Levels:        2
    Number of Tasks without Components:        0
    Number of Subtasks without Predecessors:   4
    Number of Subtasks without Successors:     2

    For each Subtask:
    Type                    Max Number  Min Number  Average Number
    Component                  12          12          12.00
    Predecessor                 4           0           1.08
    Successor                   4           0           1.08

  Part III. Resource Information
  -----------------------------
    Total Number of Defined Agents and Roles:       1
    Total Number of Defined Development Tools:      3
    Total Number of Defined Required Resources:     4
    Total Number of Defined Provided Resources:     7

    For each Activity:
    Type                    Max Number  Min Number  Average Number
    Agent                       1           1           1.00
    Tool                        1           1           1.00
    Required-Resource           3           0           0.87
    Provided-Resource           3           1           1.25

  Part IV. Activities that miss some kind of Information
  -----------------------------------------------------
    TOTAL NUMBER OF SUBTASKS WITH RESOURCES:               8
    ------------------------------------------------------
    Total Number of Subtasks without Agents:               0
    Total Number of Subtasks without Tools:                0
    Total Number of Subtasks without Required-Resources:   3
    Total Number of Subtasks without Provided-Resources:   0

NIL
KC> █
```

Figure 4: Example output from an OPM process analysis query

Further, most of these analysis functions incorporate routines for generating different types of reports (e.g., raw, filtered, abstracted, or paraphrased into structured narrative) which can be viewed interactively. Similarly, reports can be automatically generated as desktop presentation materials or documents formatted for publication. The paraphrasing function employs classic natural language generation methods that traverse and unparse a semantic network following the approach originally due to Simmons and others from the 1970s [SS72]. It also now includes routines supporting the generation of materials that instantiate report or presentation templates coded in HTML for dissemination using a corporate intranet, as we will show later.

We also must address *validating* the OPMs that we capture [O87,OG90]. Here we rely upon an iterative and incremental method for modeling, verifying, refining, and validating OPM knowledge that is acquired from different people at different times. Typically, we have found that three iterations across these activities is required to achieve an external validation sign-off from the process participants. Further, when possible to-be process alternatives are identified, we must also assess their *feasibility* given resources available or likely to become available for different business processes. Thus, we rely on our experts to review, informally modify, and sign-off on various descriptions and visualizations of the OPM that are generated by the Articulator and related utilities.

Overall, our experience has been that automated verification and participatory validation of OPMs is a great source of short-term, high-value results and insight which can be provided to a business organization through a process engineering effort.

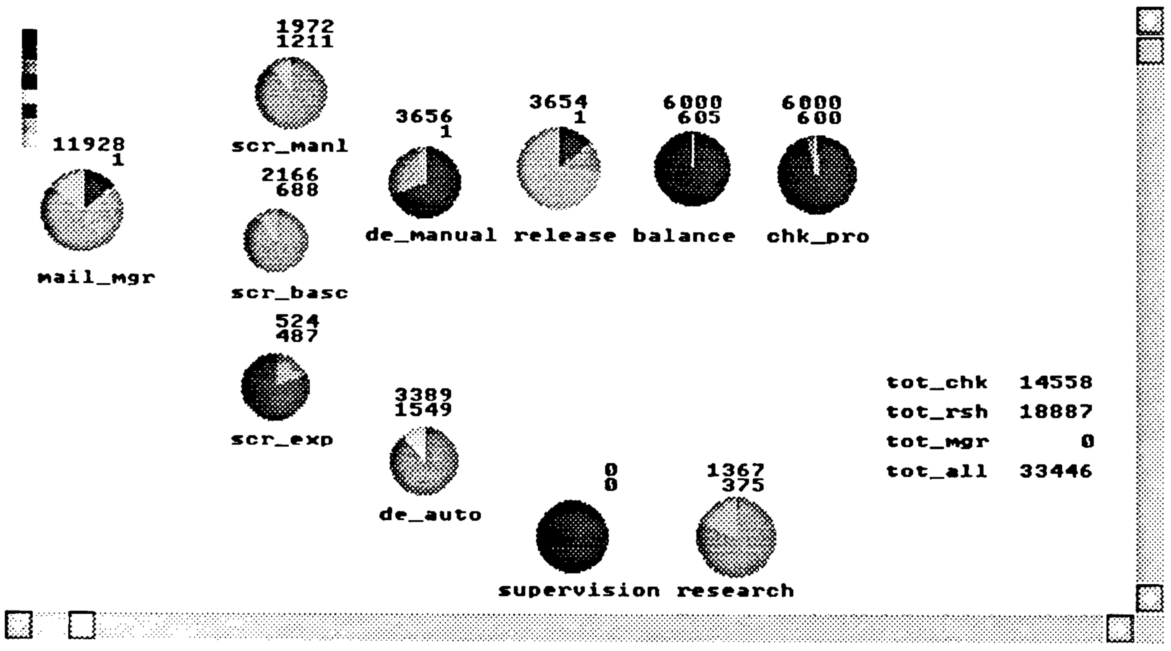
Simulation

Since process models in our scheme are computational descriptions, we can simulate or symbolically execute them using knowledge-based simulation techniques supported by the Articulator [MS90]. In simple terms, this is equivalent to saying that simulation entails the symbolic performance of process tasks by their assigned agents using the tools, systems, and resources to produce the designated products. Using the earlier example, this means that in simulating an instance of the AP OPM, Mary's agent would "execute" her management tasks according to the task precedence structure specified in the OPM instance, consuming simulated time, budgeted funds, and other resources along the way. Since tasks and other resources can be modeled at arbitrary levels of precision and detail, then the simulation makes progress as long as task pre-conditions or post-conditions are satisfied at each step (e.g., for Mary to be able to assign staff to the report production task, such staff must be available at that moment, else the simulated process stops, reports the problem, then waits for new input or command from the simulation user).

Our use of knowledge-based simulation mechanisms also support features that are uncommon in popular commericial simulation packages. For example, using symbolic execution functions, the Articulator can determine the path and flow of intermediate process state transitions in ways that can be made persistent, queried, dynamically analyzed, and reconfigured into multiple alternative scenarios. Persistence storage of simulated events enables the ability to run simulation forward and backward to any specific event or update to the knowledge base. Queries provide one such mechanism to retrieve or deduce where an event or update of interest occurs. Dynamic analysis can monitor usage or consumption of resources to help identify possible bottlenecks in OPM instances. Then, using any of these functions, it is possible to run a simulation to some point, backup to some previous context, create new OPM instance values (e.g., add more time to a schedule, more staff to a overloaded workflow, or remove unspent money from a budget), branch off a new simulation trajectory that can subsequently be made persistent, and so forth. Furthermore, we can also employ the paraphrasing and report generation functions noted above to produce narrative-like descriptions or summaries of what transpired during a given simulation run. Thus, knowledge-based simulation enables the creation and incremental evolution of a network of event trajectories which may be useful in evaluating or systemically forecasting the yield attributal to to-be process alternatives.

Simulations also allow us to dynamically analyze different samples of parameter values in OPM instances. This in turn enables the simulated processes to function like transportation networks whose volumetric flow, traffic density, and congestion bottlenecks can be assessed according to alternative (hueristic or statistical) arrival rates and service intervals. When used this way, as a classic discrete-event simulation, process experts find it easy to observe or discover process bottlenecks and optimization alternatives. Similarly, when repetitive, high-frequency processes such as AP are being studied, and when data on events and process step completion times/costs can be empirically measured or captured, then this provides a basis for assessing and validating the *replicability* of the simulated process to actual experience. As this was the situation for us during this study, we found we could achieve simulation results on as-is AP processes that were consistent with observed measurements within 85-98% for the instance value samples investigated.

Since commercially available discrete-event simulation now support animated visual displays, we employ them so that process experts can further validate as-is and to-be process simulations under different scenarios as animated displays ("business process movies"). These animated OPM simulations in turn can be modified, re-executed, and viewed like other simulations. Although we cannot conveniently show such animations in printed form, the following two snapshots captured from such an animated simulation may suggest what can be observed. In Figure 5, we have modeled an eight person activity for performing an instance of the AP process. The figure depicts the structure of the workflow, which agents currently perform what tasks, and work units-in-progress quantities (e.g., the backlog of invoices cleared, problematic invoices, and checks released).



```
tot_chk   14558
tot_rsh   18887
tot_mgr       0
tot_all   33446
```

Figure 5: Visual display from an animated multi-agent simulation

Following this, Figure 6 displays a snapshot of an accompanying pie chart depicting current workload, division of labor, and activity-based cost figures (lower right) for a simulated workflow volume.

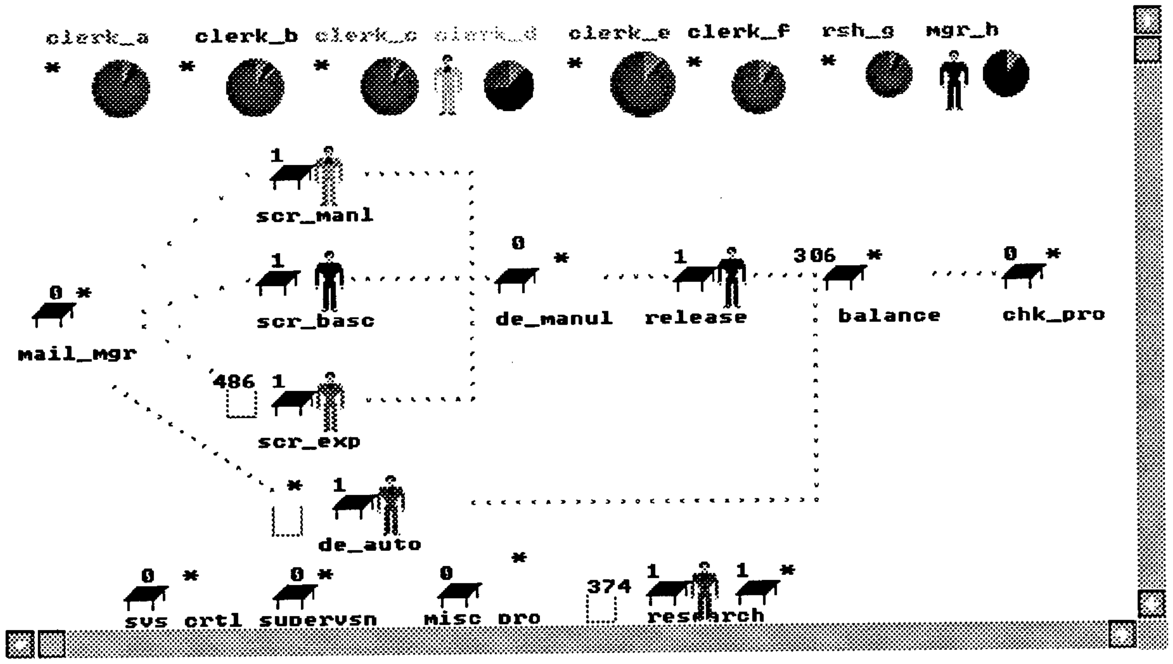


Figure 6: Visual display of dynamic pie-chart depicting current workload, division of labor, and aggregate costs

We have used the Articulator environment to model, analyze, and simulate a variety of organizational processes. In this regard, we have constructed OPMs and instances for organizations within businesses and government agencies, focused on activities involving team-based IT product design and review processes, as well as department and division-wide IT production and support processes that include tens to hundreds of participants. Such OPMs typically include dozens of classes of agents, tasks, resources, and products, but a small number of IT tools and systems, while the OPM instantiation may include 1-10+ instances of each class. Our experience to date suggests that modeling existing processes can take from 1-3 person-days to 2-3 person-months of effort, analysis routines can run in real-time or acceptable near-real-time, while simulations can take seconds to hours (even days!) depending on the complexity of the OPM, its instance space, and the amount of non-deterministic process activities being modeled. Note however that simulation performance is limited to available processing power and processor memory, thus suggesting better performance can be achieved with (clusters of) high performance computing platforms.

Overall, our experience with these simulation capabilities can be summarized according to the kind of mechanisms employed. We found that knowledge-based simulation was of greatest value in supporting exploratory analysis of OPMs where attention was focused on understanding fine-grained or deep causal relationships that could arise during a symbolic execution. This help facilitate *qualitative* insights into the operation of OPMs. In contrast, discrete-event simulation was of greatest value in validating and assessing "shallow" OPM instances using coarse-grain and statistically representative data samples. This helps facilitate *quantitative* insights into the operation of alternative OPMs. Thus, together we find that both knowledge-based and conventional simulation functions are most helpful when used to complement the strengths of one another.

Redesign

Process redesign is concerned with structural transformation of workflow or other relational properties associated with a process or sequence of process steps. At this point, most of the knowledge for how to transform a process, or what transformations are available or applicable remains informal. Most of the popular treatments on business process redesign are motivational rather than systematic and analytically reproducible.

Recent progress is beginning to suggest that when process descriptions can be represented as an attributed directed graph or semantic network, then knowledge-based techniques and computational mechanisms may be applied to identify or eliminate possible process redesigns. These efforts involve the use of rule-based representations to recognize and transforms patterns in the formal representation of a process model [KS96]. In this regard, the condition part (the left-hand side) of the rule recognizes a process pattern, while the action part (the right-hand side) invokes a method that replicates some form of a case-based, "best practice", flow optimization, or other process redesign hueristic.

Within our research group, we have been exploring process redesign in the following manner. We specify a set of measurements on graph connectivity, interrelations, and node/link complexity that may be used as indices into a taxonomy of process transformations [Ni94,Ni96]. Such metrics distill domain-independent, application domain-specific, and setting-specific instance patterns in the formal representation of a modeled process. A taxonomic classification of business process transformations can then be employed when populated with domain-independent transformations (e.g., parallelize a sequence of process steps if mutually independent); application domain-specific transformations (reduce the handling of problematic invoices in AP by pre-filtering invoices received and recycling back those with problems); and setting-specific transformations (if Mary's workload is reduced, she can perform Patrick's tasks as well, thereby freeing up Patrick to perform other tasks).

Formalizing the condition metrics patterns, as well as the action transformations, appears most attractive and most widely reusable for domain-independent process redesigns. Alternatively, formalizing application-specific metrics and transformations should yield a more powerful approach leading to more dramatically streamlined process redesigns. The price paid to acquire such knowledge is great, but common business process such as AP are almost universally found in every business. This suggests the possibility of amortizing the investment in knowledge acquisition for the chosen application domain across many possible reapplications. Finally, setting-specific metrics and transformations are probably most likely not to be codified or automated, since the cost of capturing and codifying such idiosyncractic knowledge is likely to be outweighed by the potentially short duration of its utility. However, such intimate knowledge of the setting (and participants) where process redesign is being considered is likely to be among the most influential variables that determine the success or failure of a business process redesign effort. Thus, our approach is focusing on codifying the most reusable knowledge across common business process application domains, while relying on the active engagement and participation of the people who perform the process, as well as have a stake in its redesigned outcome, to select among process redesign alternatives which can be identified and further engineered.

Downstream Process Engineering: Visualization

through Evolution

As we improve our ability to construct and redesign plausible models of different organizational processes, we have found that it is increasingly important to be able to quickly and conveniently understand the structure and dynamics of complex OPM instances. As such, we have developed a graphic user interface (GUI) for visualizing and animating OPM instances. This *process-based interface* (PBI) is coupled to another computational facility that can automatically translate OPM instances into executable process programs. These process programs are then downloaded into a program interpreter that serves as a *process execution mechanism.* In turn, the process execution mechanism and GUI enable OPM developers to prototype or enact *process-driven IT environments*. These capabilities can be used to reflect, guide, try-out, and support how users work with process-driven ITs. These capabilities are described next.

Visualization

PBI provides graphic visualizations of task precedence structure on a role-specific basis for each user (i.e., agent instance) [MS92]. Since process tasks can be modeled and hierarchically decomposed into subtasks of arbitrary depths, then PBI provides users with a subtask window and an associated workspace. Figure 7 shows an example of this presentation for the top-level view of the AP process, which highlights the process's logical workflow, from left to right. Since a subtask precedence structure appears as a directed graph, we associate a development *status* value (i.e., none, allocated, ready, active, broken, blocked, stopped, finished) with each process task or step (nodes in the graph). For ease of understanding, each of these status values is represented in the PBI display in a distinct color (not shown here), so that the current *state* of a process task can be observed as a color pattern in the direct graph. Further, as PBI also incorporates a facility for recording and replaying all changes in process task state, evolving process state histories can be maintained and visualized as an animation of changing task step status colors. Subsequently, we have found that department managers in business organizations can quickly browse such a PBI display to ascertain the current status of an arbitrarily complex production process to varying degress of detail. The interested reader should consult [MS92] and http://www.usc.edu/dept/ATRIUM/Process_Life_Cycle.html to see a number of examples.

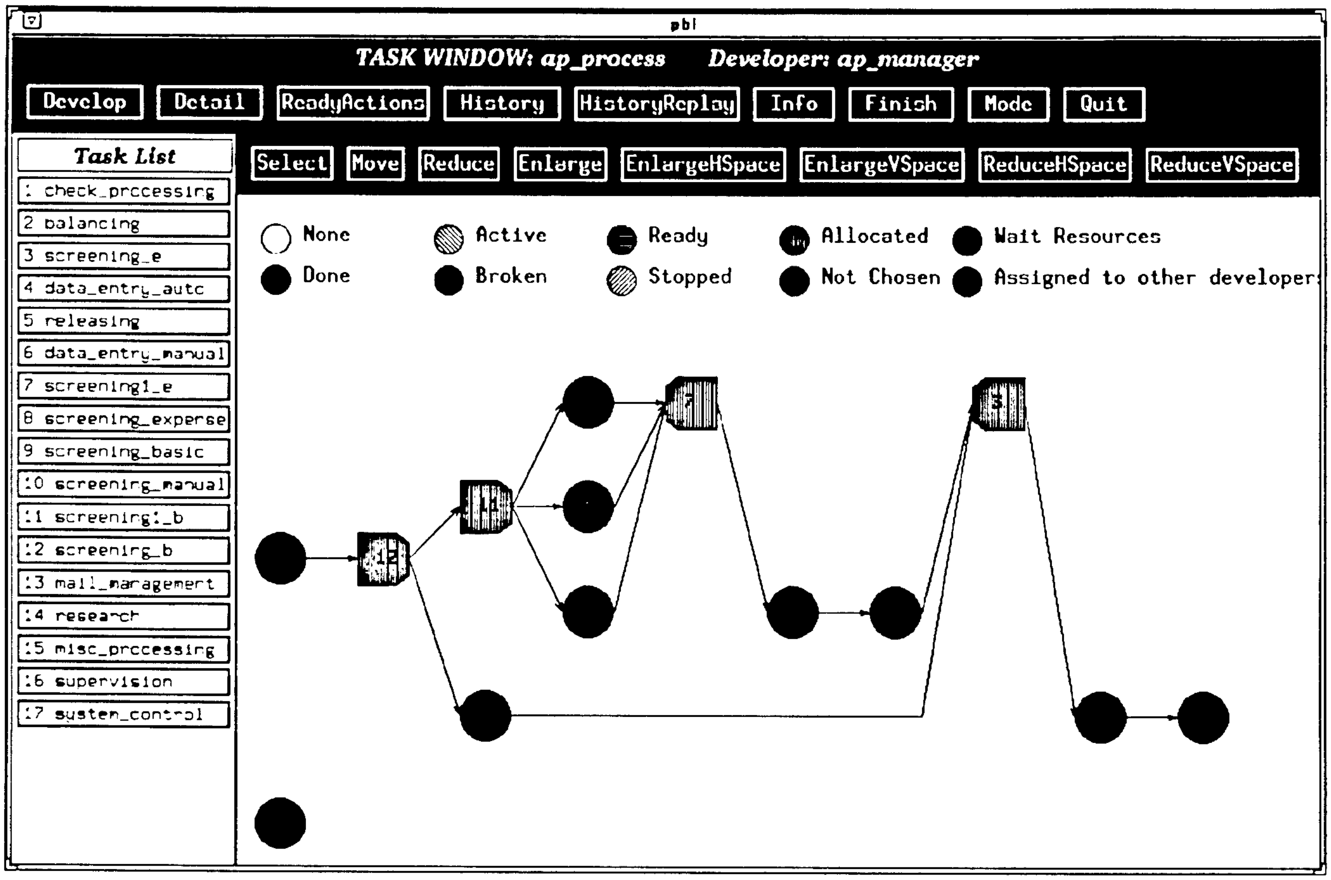


Figure 7: A Top-Level View of the Accounts Payable Workflow

Prototyping and Performance Support

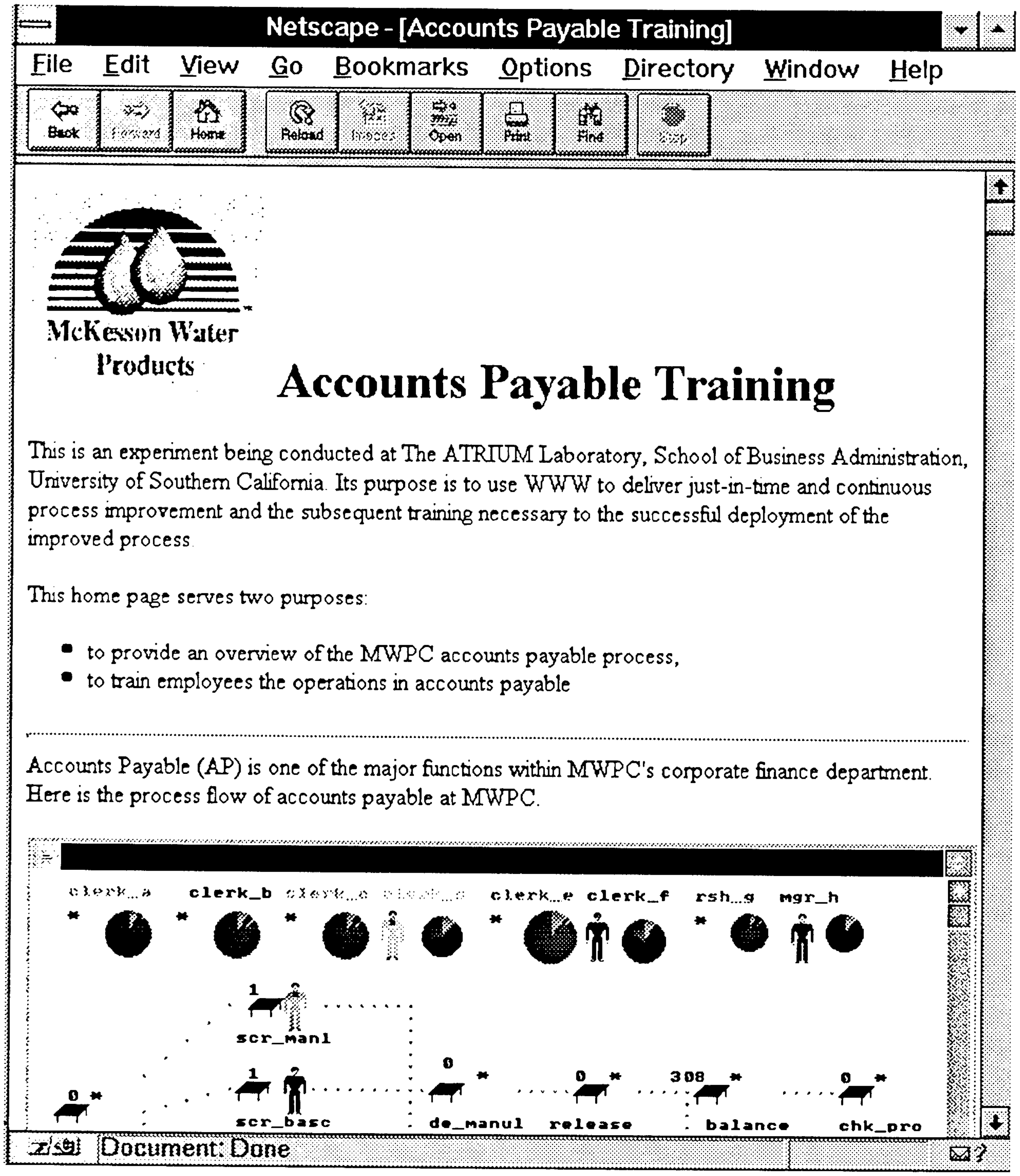
The process execution mechanism that backs PBI can also accept an OPM as its input. Since OPMs need not include instance details until process enactment, then it is possible to use these OPMs to create prototype mock-ups of process-driven environments. These prototypes show the user the look-and-feel of how the emerging process-driven environment would appear. That is, the OPM serves to provide role-specific views of process task precedence structure, which in turn guides users in their use of IT tools, systems, and data resources. This provides process users the support mechanisms and opporunity to try out or "rehearse" new ways of doing their work through process redesign alternatives. Thus, since the Articulator accomodates partially decomposed OPMs, then these OPMs can also be downloaded into the process execution mechanism to visually display and interactively walkthrough role-specific usage scenarios.

We find this ability to walkthrough, tour, or rehearse process workflow prior to commiting to its implementation extremely useful in supporting an OPM construction effort. In this way, we can support the iterative, incremental specification and refinement of OPMs in an improvement-oriented evolutionary sense. Accordingly, we have found that process prototyping is an effective enabler for eliciting user feedback. This helps to facilitate user-level understanding of their processes when supported by a process-driven computing environment. Process prototyping also provides a basis for user empowerment in controlling the design, refinement, and improvement of local processes. This helps to facilitate the adoption of redesigned business processes, since the people who perform the process can tailor them to better support and accomodate their process expertise.

We have also found that process prototypes can be sufficiently enriched to support process training and task performance. In particular, given the knowledge-based representation and processing mechanisms for upstream process engineering, we find that we can automatically generate multi-perspective views, documentary content, and tool invocations that support on-demand process training and navigation as an aid that supports process performance.

Gery [Ge91] defines an *electronic performance support systems* (EPSS) as the use of computer-based systems to provide on-demand access to integrated information, guidance, advice, assistance, training, and tools to enable high-level job performance with a minimum of support from other people. The goal of an EPSS is to provide whatever is necessary to ensure performance and learning at the moment of need. As such, we have developed a facility for generating a process-centered EPSS from knowledge-based OPMs.

We have been experimenting with the use of new paraphraser that generates views and descriptive content from OPMs represented within the Articulator's knowldedge base. Views are generated that center on the focal process flow, its description, participating organizational roles and task responsibilities, inputs and outputs, and supporting IT tools and systems. Further, we have adopted the use of descriptive templates--generic descriptions shells--that are coded in the hypertext markup language (HTML) for publication and wide-area distribution over a corporate intranet or the Internet. In this regard, this paraphraser is directed to generate certain kinds of content that populate the format coded into the description templates. Using this approach, we can generate OPM-based EPSS capabilities whose content can be delivered globally, but updated from a single point (the OPM specification). Figure 8 shows an example of the beginning of the first page of content generated for an Accounts Payable process that includes a generic prologue, and a process flow diagram produced by a process simulation tool noted earlier.


Netscape - [Accounts Payable Training]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
McKesson Water Products
clerk_a clerk_b clerk_c clerk_d clerk_e clerk_f rsh_g mgr_h
1 scr_manl
1 scr_basc
0 de_manul
0 release
308 balance
0 chk_pro
Document: Done

Accounts Payable Training

This is an experiment being conducted at The ATRIUM Laboratory, School of Business Administration, University of Southern California. Its purpose is to use WWW to deliver just-in-time and continuous process improvement and the subsequent training necessary to the successful deployment of the improved process.

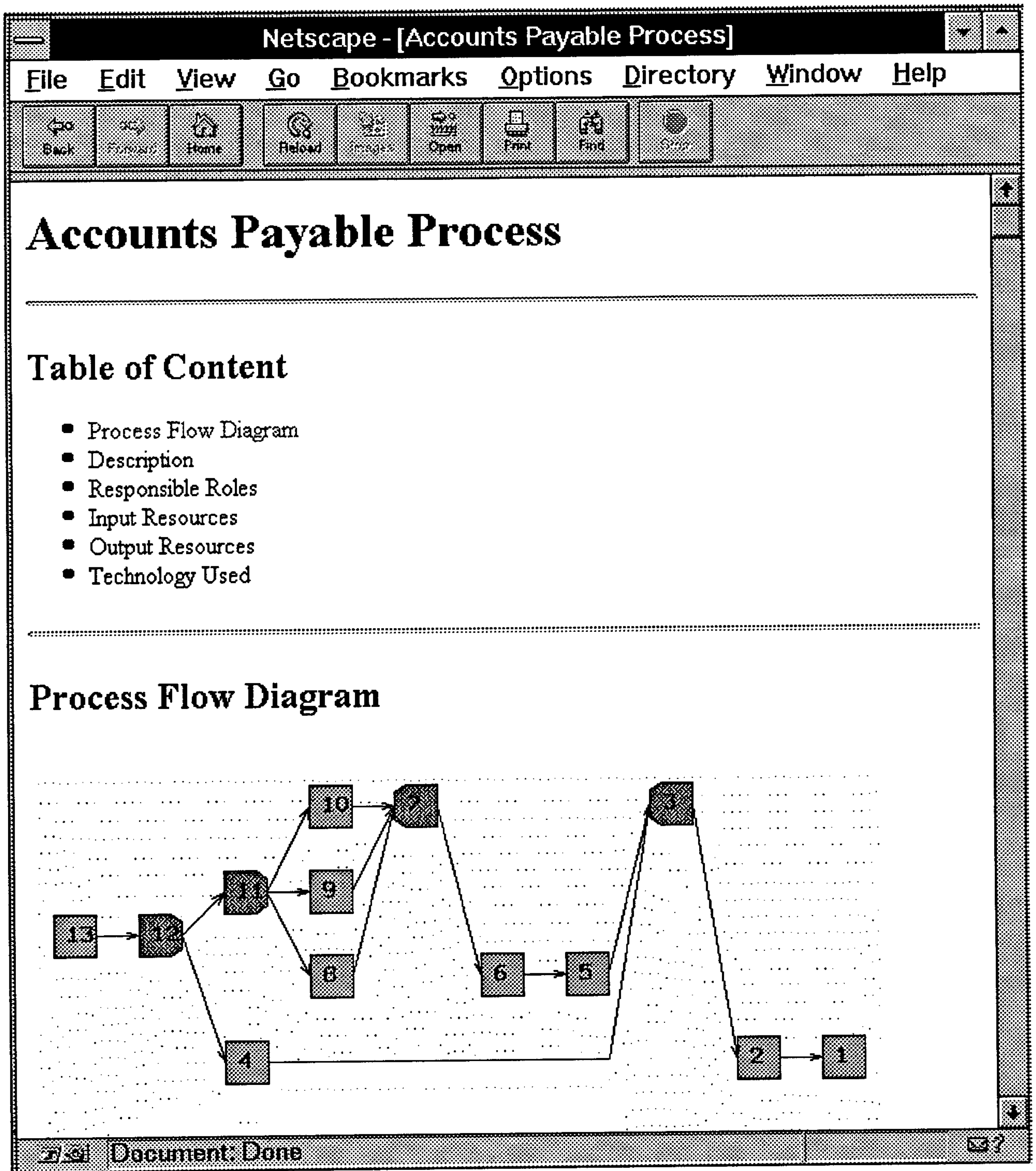
This home page serves two purposes:

- to provide an overview of the MWPC accounts payable process,
- to train employees the operations in accounts payable

Accounts Payable (AP) is one of the major functions within MWPC's corporate finance department. Here is the process flow of accounts payable at MWPC.

Figure 8: Intranet-based EPSS content generated from an Accounts Payable Process Model

Similarly, other pages from a generated web of performance support materials can be produced from a paraphrase of an OPM. Figure 9 shows another page that follows from links (not shown) on the page in Figure 8.


Netscape - [Accounts Payable Process]
File Edit View Go Bookmarks Options Directory Window Help
Back Forward Home Reload Images Open Print Find Stop
Document: Done

Accounts Payable Process

Table of Content

- Process Flow Diagram
- Description
- Responsible Roles
- Input Resources
- Output Resources
- Technology Used

Process Flow Diagram

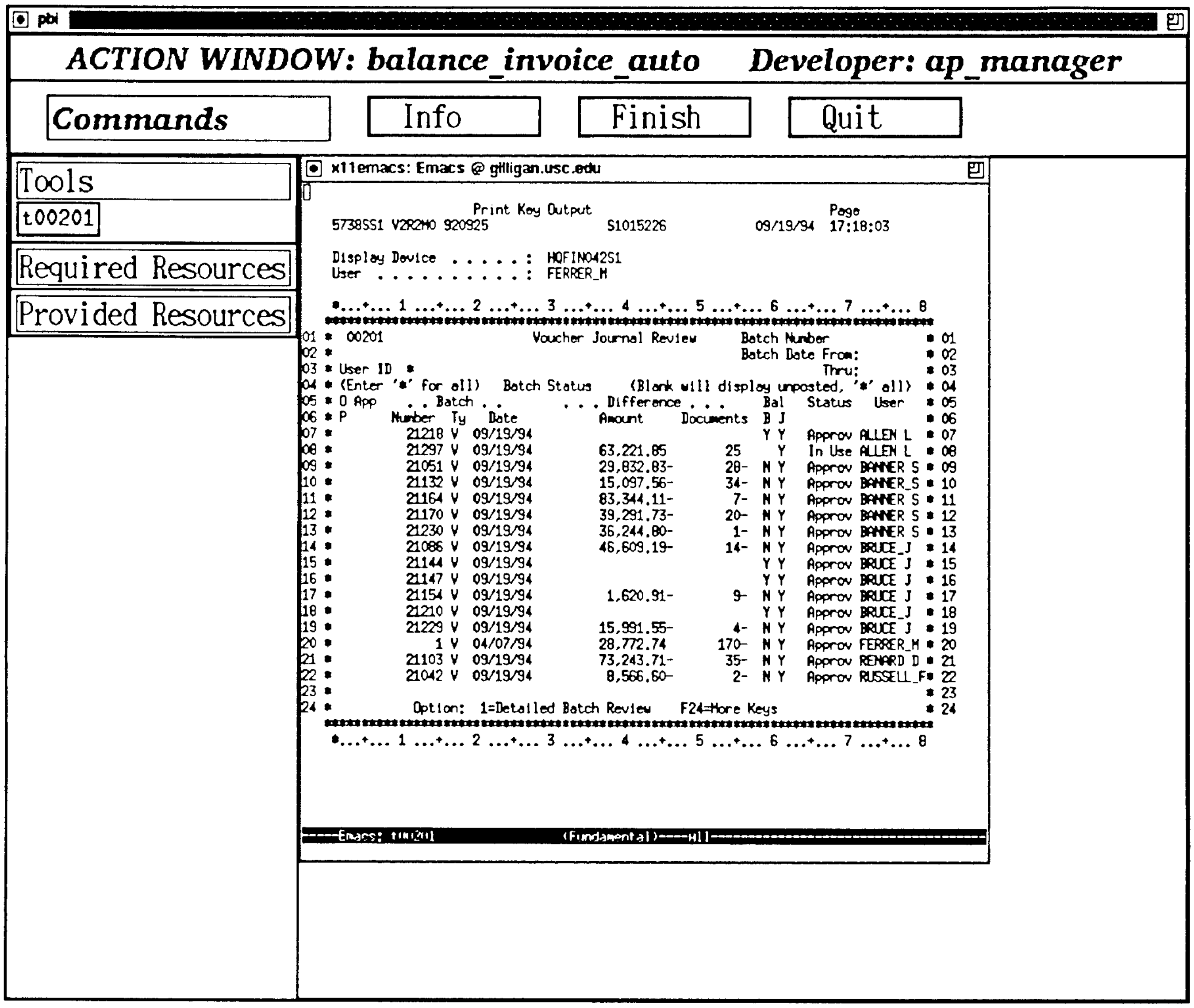
13
12
11
10
9
8
7
4
6
5
3
2
1

Figure 9: A later page in a web of performance support content generated from an Accounts Payable Process Model

Finally, it is worth noting that the ability to accomodate end-users or process performers to update the performance support content corresponds to their ability to modify the OPM specification. This in turn represents an ability to provide support for user-directed *dynamic* refinement and on-demand generation of performance materials [LAF95], which can further facilitate the learning, codification, and ownership of corporate process knowledge [SZ95].

Integration, Enactment, and History Capture

The process execution mechanism and PBI provides IT tools, systems and associated data resources (e.g., objects, files, databases, spreadsheets) which are served to users so that they can perform their work. We refer to this capability as *process enactment*, meaning that users can perform or enact the modeled process tasks, subtasks, or actions assigned to them using the IT tools, systems, and data resources delivered to them at their displays and fingertips when needed. Figure 10 shows an example of a process enactment view, which provides the IT applications, tools, data objects and workspace appropriate for the user assigned to this order fulfillment process action.


pbi
ACTION WINDOW: balance_invoice_auto   Developer: ap_manager
Commands
Info
Finish
Quit
Tools
t00201
Required Resources
Provided Resources
x11emacs: Emacs @ gilligan.usc.edu
----Emacs: t00201   (Fundamental)----All----

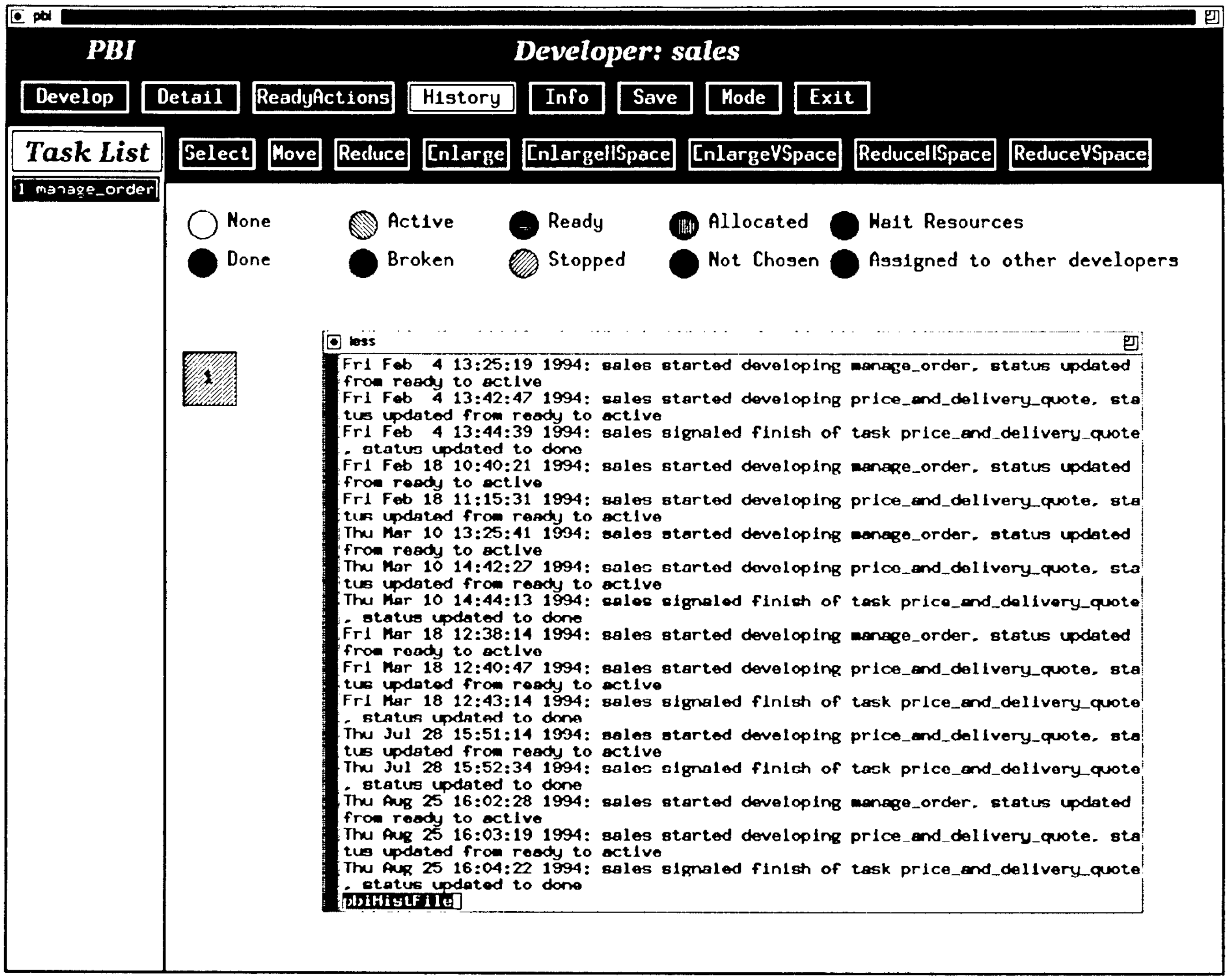
```
                     Print Key Output                               Page
 5738SS1 V2R2M0 920925                 S1015226            09/19/94  17:18:03

 Display Device . . . . . :  HQFIN042S1
 User . . . . . . . . . . :  FERRER_M

 *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
 ********************************************************************************
01 *  00201                          Voucher Journal Review      Batch Number        * 01
02 *                                                             Batch Date From:    * 02
03 * User ID  *                                                             Thru:    * 03
04 * (Enter '*' for all)   Batch Status      (Blank will display unposted, '*' all)  * 04
05 * O App     . . Batch . .          . . . Difference . . .     Bal   Status   User  * 05
06 * P        Number  Ty   Date              Amount     Documents  B J                 * 06
07 *            21218 V  09/19/94                                 Y Y   Approv ALLEN L  * 07
08 *            21297 V  09/19/94          63,221.85          25     Y   In Use ALLEN L  * 08
09 *            21051 V  09/19/94          29,832.83-         28-  N Y   Approv BANNER S * 09
10 *            21132 V  09/19/94          15,097.56-         34-  N Y   Approv BANNER_S * 10
11 *            21164 V  09/19/94          83,344.11-          7-  N Y   Approv BANNER S * 11
12 *            21170 V  09/19/94          39,291.73-         20-  N Y   Approv BANNER S * 12
13 *            21230 V  09/19/94          36,244.80-          1-  N Y   Approv BANNER S * 13
14 *            21086 V  09/19/94          46,609.19-         14-  N Y   Approv BRUCE_J  * 14
15 *            21144 V  09/19/94                                 Y Y   Approv BRUCE J  * 15
16 *            21147 V  09/19/94                                 Y Y   Approv BRUCE J  * 16
17 *            21154 V  09/19/94           1,620.91-          9-  N Y   Approv BRUCE J  * 17
18 *            21210 V  09/19/94                                 Y Y   Approv BRUCE_J  * 18
19 *            21229 V  09/19/94          15,991.55-          4-  N Y   Approv BRUCE J  * 19
20 *                1 V  04/07/94          28,772.74         170-  N Y   Approv FERRER_M * 20
21 *            21103 V  09/19/94          73,243.71-         35-  N Y   Approv RENARD D * 21
22 *            21042 V  09/19/94           8,566.60-          2-  N Y   Approv RUSSELL_F* 22
23 *                                                                                  * 23
24 *           Option:  1=Detailed Batch Review     F24=More Keys                     * 24
 ********************************************************************************
 *...+... 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ...+... 8
```

Figure 10: A Lowest Level Action Workspace within the Accounts Payable Process Model

Process enactment is also a computational activity performed by the process execution engine. It interprets an OPM instance as its input. Thus, the OPM instance output from the Articulator represents a process enactment specification that is coded in something similar to an object-oriented operating system scripting language, or what others have called a *process programming language* [Ost87]. In this sense, our process programs are automatically derived from the process model specification by way of a special-purpose application generator [MS92,KS93]. Accordingly, the process enactment specification can incorporate any operating system command, system invocation script, or network access protocols to heterogeneous information repositories [NS91]. This means it is possible for users to perform complex information processing tasks through a process-based interface that integrates access to local/networked data resources and IT tools/systems through a common GUI presentation [MS92].

Process guided work can seem onerous if the process forces people to do their work in an awkward, rigid, or unnatural manner. People can differ in the skill, knowledge, and prior experience in performing a process. Thus, we should not unnecessarily encumber an expert in a process by requiring her to follow the same sequence of process steps needed to guide and familiarize a newly assigned process novice. As a result, we have implemented a process guidance "mode" variable that enables different levels of process guidance or enforcement to be followed. In this way, we have provided a mechanism for people in an organization, rather than us, to determine the policy for who needs to follow or conform to process guidance. We have found that this form of flexibility can serve as another mechanism to improve the acceptability and satisfaction with process guided support systems. Alternatively, it is also a way of expressing a lesson we learned for how to make process enactment more accomodating to concerned process participants.

Finally, as process enactment can provide computer-supported guidance of work tasks, we have also incorporated a facility for keeping a history of what process steps were taken, in what order, and with what results. Such a facility can serve as a record or audit trail for processes when conformance to standards (e.g., adhering to financial controls) is needed. Figure 11 displays such a trace of process events for a "sales" person performing part of an order fulfillment process. Furthermore, as this history record details who did what step at what time with what result (status update), we provided a PBI-based utility that allows the history to be replayed as a graphic animation of process flowgraph, such as that displayed in Figure 7. Similarly, the history record can serve as input to the simulation mechanisms described earlier. In both cases, this historical record serves to help gain insight for understanding how a modeled process instance got performed, which in turn can serve to support continuous process improvement efforts.



```
Fri Feb  4 13:25:19 1994: sales started developing manage_order, status updated
from ready to active
Fri Feb  4 13:42:47 1994: sales started developing price_and_delivery_quote, sta
tus updated from ready to active
Fri Feb  4 13:44:39 1994: sales signaled finish of task price_and_delivery_quote
, status updated to done
Fri Feb 18 10:40:21 1994: sales started developing manage_order, status updated
from ready to active
Fri Feb 18 11:15:31 1994: sales started developing price_and_delivery_quote, sta
tus updated from ready to active
Thu Mar 10 13:25:41 1994: sales started developing manage_order, status updated
from ready to active
Thu Mar 10 14:42:27 1994: sales started developing price_and_delivery_quote, sta
tus updated from ready to active
Thu Mar 10 14:44:13 1994: sales signaled finish of task price_and_delivery_quote
, status updated to done
Fri Mar 18 12:38:14 1994: sales started developing manage_order, status updated
from ready to active
Fri Mar 18 12:40:47 1994: sales started developing price_and_delivery_quote, sta
tus updated from ready to active
Fri Mar 18 12:43:14 1994: sales signaled finish of task price_and_delivery_quote
, status updated to done
Thu Jul 28 15:51:14 1994: sales started developing price_and_delivery_quote, sta
tus updated from ready to active
Thu Jul 28 15:52:34 1994: sales signaled finish of task price_and_delivery_quote
, status updated to done
Thu Aug 25 16:02:28 1994: sales started developing manage_order, status updated
from ready to active
Thu Aug 25 16:03:19 1994: sales started developing price_and_delivery_quote, sta
tus updated from ready to active
Thu Aug 25 16:04:22 1994: sales signaled finish of task price_and_delivery_quote
, status updated to done
pbiHistFile
```

Figure 11: An Historical Record of Process Enactment Events

As such, we have prototyped and demonstrated a number of process-driven environments in different business and government application domains that incorporate commercial off-the-shelf systems, internally developed systems, and prototype research software technologies that can operate over local-area and wide-area networks [NS91].

Articulation: Repairing Processes that Unexpectedly Breakdown

If one wonders why we chose to call our process engineering environment the Articulator, it is due to our deep seated interested in articulation work. *Articulation work* refers to the activities people at work engage in when routine processes do not conform to the routine [BS87,Su87,MS91]. This is when business processes activities breakdown or fail to apply. As a result of such dilemmas, people make circumstantial accomodations or negotiate with others for assistance and adjustments. Articulation work then entails the effort to figure out what's gone awry, and how to identify and implement ways to fix-up, work around, or hand-off the dilemma at hand.

In open-ended real-world work settings, articulation work is common and widespread. Even a highly repetitive business process, such as the processing of invoices and issuing checks within an AP process, is never an activity that is repeated in an identical manner each time it is enacted and performed. Idiosyncratic circumstances, contingencies, and the movement of people and resources through time and space may account for this. However, it is a mistake to believe that business processes that involve people can be structured and constrained to be repetitive without variation. Thus, handling exceptions, special cases, and other contingencies, whether anticipated (i.e., as low probability events) or unanticipated, is in some sense a normal part of business processes and work. Logically robust but rigid computational procedures or other mechanistic means that disregard or ignore articulation work are doomed to fail, or to make adaptive work efforts troublesome. Accordingly, we have sought to develop an alternative to rigid automated process enactment mechanisms that recognizes and seeks to support articulation work.

We have developed a KB approach to supporting process articulation work [MS91,Mi92,MS93]. Although our efforts initially focused on activities in the domain of software engineering, we sought to develop representation scheme and processing mechanism that treated process articulation as a domain-independent phenomenom. In this regard, our approach to support process articulation using the Articulator focuses on a three-phase computation involving diagnostic classification, replanning, and rescheduling [MS93]. In our approach, we assume that business processes can be formally modeled then simulated or enacted using an Articulator-based environment. As the process model or model instance accounts for the logical representation of the process, and not all the circumstances and resource configurations at hand in the workplace at any moment, then contingencies can arise that are not part of the planned process instance. Thus, the enactment of the process instance will in some way not be effective. As all entities represented in the Articulator's KB are a sub-class of the root class of "resource", then any problem that the Articulator can possibly detect or reason about must be bound to some type of resource. Subsequently, in the Articulator's resource-based ontology, different classes of resource failures or breakdowns are the root cause of articulation work. Therefore, when a process enactment gets into trouble, it is due to a resource-based problem and causality. If the resources are those directly involved in enacting the process (people, IT tools and systems, task inputs and outputs, etc.), then it may be possible to detect their breakdown, to seek courses of action leading to a repair, and to resume the process instance if successfully repaired.

When a person (or simulated agent) encounters a process resource breakdown, the Articulator can be provided with the *context* of the problem at hand. This includes the model of the process being performed, the history of what steps have been completed so far, the resources (classes and instance values) involved in the process's enactment, and the schedule (if any) for completing the remaining process steps. The Articulator then performs a diagnostic classification of the resource breakdown, seeking to match with one or more of the known classes of process breakdowns that we have categorized through various empirical studies and abstraction [MS91]. A view of this taxonomy is shown in Figure 12.

As a given breakdown or failure may not have a single cause, then the Articulator's breakdown classifier may find one or more possible classes of failure. In turn, each classified breakdown returns an index value which then guides processing in the second phase, the search for a root cause (or causes) of the breakdown. Again one or more causes may be found for each instance of a breakdown. As causes are surfaced, then another search begins to find possible ways to repair the process instance so that work can in some way continue. Repair hueristics, also derived from empirical study, are searched, and the results that match are returned in a partial order. These repairs look for information represented in the process enactment context to see what resources (e.g., other people, tools, alternative inputs, etc.) are nearby which may be available for use to repair the breakdown. If a repair is found and integrated, process enactment can resume, or be modified to accomodate the circumstances. Repairs can entail modifying the process instance by reconfiguring task flow by gaining access to other resources, or by handing off the problem to someone else and defer completion of this process enactment instance.

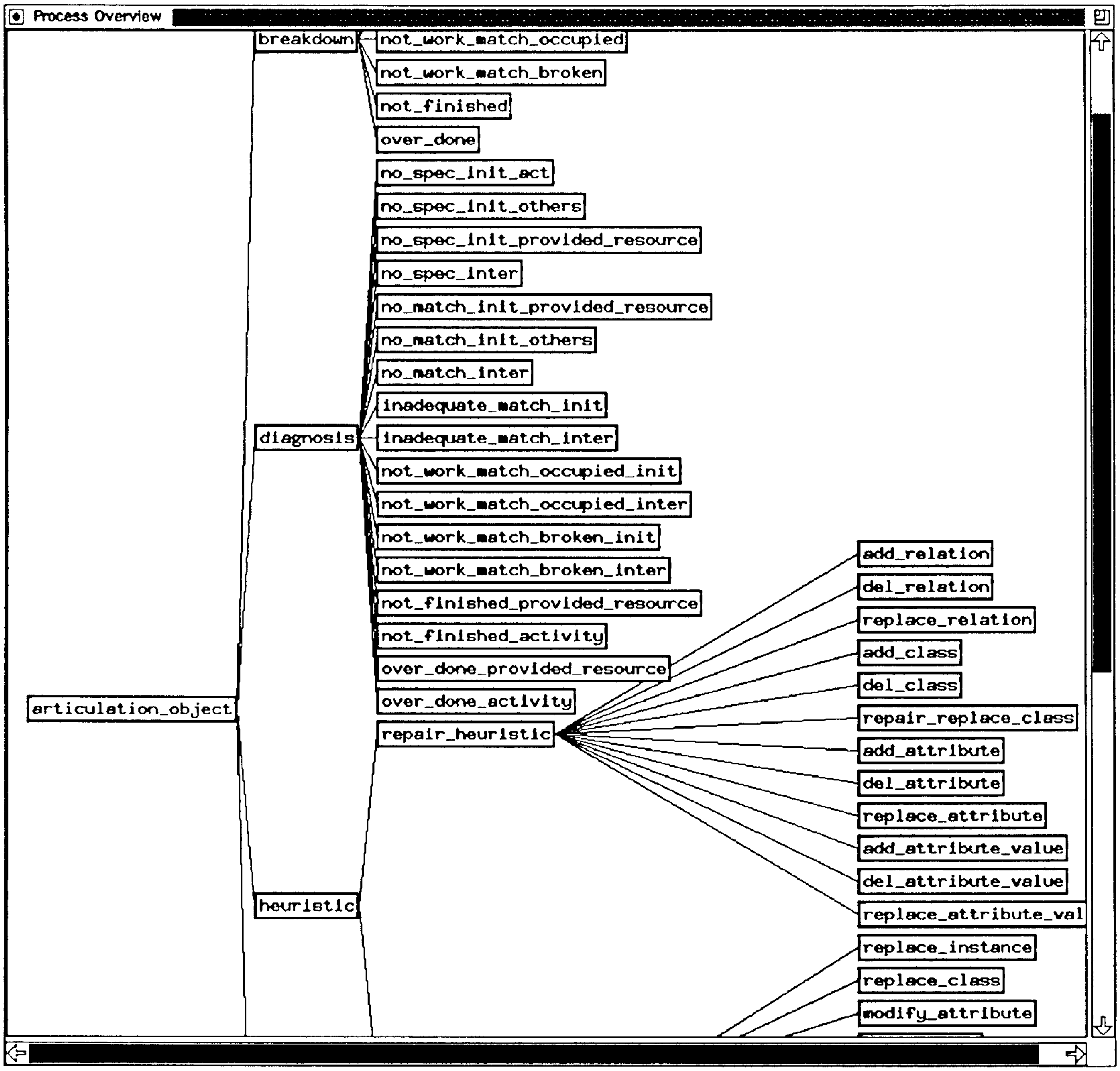


Figure 12: A Partial View of a Classification Taxonomy for Articulation Breadowns, Diagnoses, and Repair Hueristics

Next, the choice for which repairs to make involves selecting from the alternatives that have been found.

In the Articulator, this form of replanning involves evaluation of both local (context-specific) and global constraints that guide the selection of repairs. Such evaluation may in turn evaluate constraints imposed by a schedule or completion time target for the process enactment. In this way, process instance repair involves the interaction of replanning and rescheduling rule-bases, each of which includes between 150-200 rules for this purpose. After this computation cycle completes, a modified version of the remaining process enactment instance is produced.

Finally, if a suitable repair can be found, replanned and rescheduled, we face a final dilemma--that of determining whether the root cause of the breakdown or failure was in fact circumstantial (specific to this enactment instance) or systemic (specific to the process model). Our experience has been that this is still an empirical issue, requiring observations of recurrence or lack thereof. Accordingly, we use these articulation support mechanisms to suggest and repair process enactment instances under user control, and record their occurence in a process repository for later evaluation. This in turn also serves to provide a basis for improvements to the process model. Nonetheless, the study and development of computational mechanisms needed to understand and support articulation work, as well as process enactment repair, remains a rich area for further research.

Other Advanced KBPEE Technologies

In addition to the the computational mechanisms described so far, our approach utilizes mechanisms not described here. These include mechanisms for:

- formalization of the knowledge representation meta-model [MS96] and product model [CS96],
- process scheduling and administration [Mi92],
- knowledge-based repository for process model assets [ML92],
- process-driven information systems engineering and re-engineering environment [CS91,MS92].

Thus, we believe our approach can allow us to construct and demonstrate a computational framework for modeling, enacting, and integrating team-oriented process-driven work environments for redesigned business organizations. As such, we are now working with our research sponsors to prototype and demonstrate a small number of process-driven environments in different business domains that incorporate commercial off-the-shelf systems, internally developed systems, and prototype research mechanisms, all operating on Unix and PC workstations over local-area and wide-area networks.

Comparison with Other Ongoing Research

Much of the research in engineering process descriptions which influence our work at the USC ATRIUM Laboratory focuses on representations of complex organizational processes and architectures for process-centered application support environments. Our earliest efforts focused on the processes associated with the engineering of software systems [cf. ST96]. One of the insights we gained from these experiences was that we are likely to acheive the most dramatic results and payback from a process engineering effort when focussed on processes that (a) have a high rate of instantiation (are performed frequently), and (b) have relatively short process cycle times. Some software engineering processes fall into this category (e.g., software reviews, inspections, configuration management, and software testing), while others do not (software requirements analysis and system design). This insight perhaps foreshadows where significant gains and favorable return on investments in process reengineering efforts are most likely to be realized when using a process life cycle engineering approach. Thus, when selecting opportunities for process improvement or reengineering in an application domain such as corporate financial operations, high frequency and short duration processes, such as those found in the handling of accounts payable and account receivable, are prime candidates, whereas processes associated with quarterly or annual book closings would not.

Over the past five or so years, we have broadened our focus to business process domains, such as corporate finance, military procurement, electronic commerce, and supply chain logistics. Much of the early research into process engineering can be attributed to efforts focused on processes for software engineering [CK92,SM93,HB94]. However, the focus is broadening to include other business processes and workflow technologies [SG96]. Nonetheless, we will focus our attention to those efforts in process engineering that employ a knowledge-based approach, without restriction to application domain.

Work by Jarke, Mylopoulos, Yu and colleagues [JJ90,YM94,YM96,YML96] have developed knowledge-based representations for modeling and reasoning about complex business processes. Their approaches are similar in spirit to our approach to process meta-modeling, modeling, and analysis. However, process simulation or execution activities have not yet been addressed. The PSS project in England [BPe91] developed an approach to process modeling that also support enactment language based on an object-oriented knowledge representation notation, as we have done as well. The Grapple system [HL88], on the other hand, relies on a set of goal operators and a planning mechanism to represent process enactment operations. These are used to demonstrate goal-directed reasoning about software engineering processes during modeling and enactment. While these efforts lack support for upstream process engineering, the Articulator lacks support for generative process planning. However, the Articulator does provide a replanning mechanism for repairing process instance enactments that breakdown.

The AP5 environment [BN93], developed at the USC Information Sciences Institute, and the Marvel environment [KF87] and successors developed at Columbia, use pattern-directed inference rules to model and trigger software engineering process actions during process enactment. AP5 has an implementation of the Articulator process meta-model, which was used to experiement with the integration of heterogeneous process enactment environments. The SMART environment developed at Hewlett-Packard Laboratories and USC also successfully undertook a similar experiement [GM94]. Marvel supports the creation of process models through rule-chaining which trigger automated events or procedures. However, its strength lies in its ability to support rule-based reasoning and generation of process integration parameter values that then provide reactive guidance for process enactment [HK92]. In contrast, the Articulator and SMART only provides a mechanism for generating and integrating proactive process-driven application development or execution environments.

Beyond these efforts, work by Selfridge and Terveen [ST96] and Stein and Zwass [SZ95] draw attention to the need to provide supporting mechanisms for the capture, management, and update of the *knowledge and learning* associated with business process performance and expertise. The acquisition and employment of this knowledge can be important when redesigning process structure or flow. People who perform mundane, arcane, or creative business processes as part of their work often possess deep knowledge about the intracacies, weaknesses, and failings of their processes [Su87]. Cavalier disregard of this knowledge and expertise in process redesign efforts is a likely contributor to recurring failure of such efforts. Similarly, getting process participants to understand and learn how to perform their work in a redesigned process requires more than simply showing them process visualizations or providing them with nominal training seminars. As learning scientists such as Roger Schank have found [Sch94], people learn in different kinds of ways requiring different kinds of "learning architectures" to support their learning. These architectures, when realized as computer-based support environments, should provide modalities for learning through modeling, analysis, simulation, rehearsal, performance support, enactment, and articulation. Thus, we have also sought to support knowledge management and learning through our approach to process life cycle engineering.

In sum, no other process engineering environment today supports the full process life cycle. However, we have investigated and demonstrated supporting mechanisms for each of the process life cycle activities described earlier. Similarly, while our focus is targeted at engineering organizational processes, our approach can be applied to both complex technical domains (e.g, large-scale software engineering, electronic design automation, agile manufacturing) and to conventional business processes (new product development, corporate finance, business planning, etc.), albeit in a radically innovative way [Dav93].

Conclusion

This paper provides a brief introduction to our approach and computational mechanisms to modeling, simulating, and integrating organizational processes that involve IT tools, systems, and data resources. These include a knowledge-based environment for re-engineering complex organization processes, and other facilities for realizing and executing these processes. We are using our results to help redesign existing organizational processes that employ large teams, and provide a coherent, scalable IT infrastructure for enacting and integrating IT-based organizational processes. Thus, we believe our approach allows us to construct and demonstrate a knowledge-based process engineering environment for modeling, enacting, and integrating team-oriented process-driven IT-based work environments for redesigned business and government organizations. Furthermore, we believe this approach can be used by others to produce similar results, especially when effort is directed toward the goal of capturing, managing, and creating value out of the knowledge that process participants bring to their work when performing business processes.

We also sought to convey where we believe the process engineering effort should focus their efforts in order to realize the greatest return for the least amount of effort. We have consistently and repeatedly received positive feedback from our corporate sponsors on the subjective value and eye-opening insights they have experienced as a result of the application of our approach to the engineering of their selected processes. When the costs and benefits have been quantified and systematically measured (for example, as depicted in Figure 6), we could justify or compare the value of alternative process redesigns. Similarly, when the expected gains are rolled up to annual numbers, our experience has been that some of our corporate sponsors attribute 5-7 figure annual returns to processes or operations that we helped to engineer using the tools, techniques, and concepts described here (cf. [Ba94]).

In closing, we recommend readers interested in an up-to-date view of ongoing research described in this paper to examine an interactive presentation found at http://www.usc.edu/dept/ATRIUM/Process_Life_Cycle.html on the World Wide Web for further details and examples.

Acknowledgements

This work is supported as part of the ATRIUM Laboratory at USC. Recent sponsors include Andersen

Consulting's Center for Strategic Technology and Research (CSTaR), AT&T Bell Laboratories, EDS, Hewlett-Packard, IBM Canada Ltd., McKesson Water Products Company, Northrop-Grumman Corporation, Office of Naval Research (contract number N00014-94-1-0889), and the Center for Service Excellence (CSE) in the USC School of Business Administration. However, no endorsements are implied. In addition, a number of people contributed to the ideas or system development work on which this report is based including Song C. Choi, Pankaj Garg, Anthony Karrer, Ming-Jun Lee, Jinhui Luo, Mark Nissen and John Noll. We also want to thank Buzz Adams, Chief Financial Officier at McKesson Water Products Company at the time of this study, for his encouragement and support of this study. All of these contributions are greatly appreciated.

References

Ba94 D. Bartholomew. Keeping water flowing in L.A.: how McKesson met customer demand after the quake. *Information Week*, Number 463, 41-43, 14 February 1994.

BN93
R. Balzer and K. Narayanaswamy. Mechanisms for Generic Process Support. In *Proc. First ACM SIGSOFT Symp. Foundations Software Engineering*, 9-20. ACM, Software Engineering Notes, Vol. 18(5), December 1993.

BS87 S. Bendifallah and W. Scacchi. Understanding Software Maintenance Work. *IEEE Trans. Software Engineering*. 13(3):311-323, 1987.

Boa93
B.H. Boar. *The Art of Strategic Planning for Information Technology: Crafting Strategy for the 90s*. John Wiley and Sons, New York, 1993.

BP91 R.F. Bruynooghe, J.M. Parker, and others. PSS: A System for Process Enactment. In *Proc. of the 1st International Conference on the Software Process*, 128-141, Redondo Beach, CA, Oct 1991.

CK92
B. Curtis, M. Kellner, and J. Over. Process Modeling. *Communications ACM*, 35(9):75-90, 1992.

CS91 S.C. Choi and W. Scacchi. SOFTMAN: An Environment for Forward and Reverse CASE. *Information and Software Technology*, 33(9), Nov. 1991.

CS96 S.C. Choi and W. Scacchi. Assuring the Structural Correctness of Software Life Cycle Descriptions. *(submitted for publication)*, 1996.

Dav93
T. Davenport. *Process Innovation: Reengineering Business Processes through Information Technology*. Harvard Business School Press, Cambridge, MA, 1993.

DJ93 V. Dhar and M. Jarke. On Modeling Processes. *Decision Support Systems*, 9(1):39-49, 1993.

EC94
Federal Electronic Commerce Acquisition Management Program Office. Streamlining Procurement through Electronic Commerce, National Institute of Standards and Technology, Washington, DC, October 1994. http://snad.ncsl.nist.gov/dartg/edi/arch.html

GS89
P.K. Garg and W. Scacchi. ISHYS: Designing Intelligent Software Hypertext Systems. *IEEE Expert*, 4(3):52-63, 1989.

GM94
P.K. Garg, P. Mi, T. Pham, W. Scacchi, and G. Thunquest. The SMART Approach to Software Process Engineering. *Proc. 16th. Intern. Conf. Software Engineering*, Sorrento, Italy, 341-350, 1994. Also appears in *Process-Centered Software Engineering Environments*, P.K. Garg and M. Jazayeri (eds.), IEEE Computer Society, pp. 131-140, (1996)

Ge91 G. Gery. *Electronic Performance Support Systems.* Ziff Institute, Cambridge, MA 1991.

Gr91 R.M. Grant. The Resource-Based Theory of Competitive Advantage: Implications for Strategy Formulation. *California Management Review*, 33(3):114-135, 1991.

HB94
G. Heineman, J.E. Botsford, G. Caldiera, G.E. Kaiser, M.I. Kellner, and N.H. Madhavji. Emerging Technologies that Support a Software Process Life Cycle. *IBM Systems J.*, 32(3):501-529, 1994.

HK92
G. Heineman, G.E. Kaiser, N. Barghouti, and I.Z Ben-Shaul. Rule chaining in Marvel: dynamic binding of parameters. *IEEE Expert*, 7(6):26-34 , December 1992.

HL88
K.E. Huff and V.R. Lesser. A Plan-Based Intelligent Assistant That Supports the Process of Programming. *ACM SIGSOFT Software Engineering Notes*, 13:97-106, Nov 1988.

JJ90 M. Jarke, M. Juesfeld, and T. Rose. A Software Process Data Model for Knowledge Engineering in Information Systems. *Information Systems*, 15(1):86-115, 1990.

KS93
A. Karrer and W. Scacchi. Meta-Environments for Software Production. *International Journal of Software Engineering and Knowledge Engineering*, 3(1):139-162, 1993. Reprinted in *Advances in Software Engineering and Knowledge Engineering*, D. Hurley (ed.), Volume 4, 37-70, 1995.

KF87
G.E. Kaiser and P. Feiler. An architecture for intelligent assistance in software development. In *Proc. of the 9th International Conference on Software Engineering*, 180-187, Monterey, CA, Apr 1987.

KS82
R. Kling and W. Scacchi. The Web of Computing: Computer technology as social organization. In M. Yovits, editor, *Advances in Computers, Volume 21*, 3-90. Academic Press, New York, 1982.

KS96
S. Ku, Y.-H. Suh, and G. Tecuci. Building an Intelligent Business Process Reengineering System: A Case-Based Approach. *Intelligent Systems in Accounting, Finance, and Management*, 5(1):25-39, 1996.

Laf95
J. Laffey. Dynamism in Electronic Performance Support Systems. *Performance Improvement Quarterly*, 8(1):31-46, 1995.

LA94
F. Leymann and W. Altenhuber. Managing Buiness Processes as an Information Resource. *IBM Systems J.*, 33(2):326-348, 1994.

Mi92 P. Mi. *Modeling and Analyzing the Software Process and Process Breakdowns.* Ph.D. dissertation, Computer Science Dept., University of Southern California, Los Angeles, CA, September 1992.

ML92
P. Mi, M. Lee, and W. Scacchi. A Knowledge-based Software Process Library for Process-driven Software Development. In *Proc. 7th Knowledge-Based Software Engineering Conference*, McLean, VA, September 1992.

MS90
P. Mi and W. Scacchi. A Knowledge-based Environment for Modeling and Simulating Software Engineering Processes. *IEEE Trans. on Knowledge and Data Engineering*, 2(3):283-294, Sept 1990. Also appears in *Nikkei Artificial Intelligence* (in Japanese), 24(1):176-191, January 1991.

MS91
P. Mi and W. Scacchi. Modeling Articulation Work in Software Engineering Processes. *Proc. of*

the 1st International Conference on the Software Process, 188-201, Oct 1991.

MS92
P. Mi and W. Scacchi. Process Integration in CASE Environments. *IEEE Software*, 9(2):45-53, March 1992. Also appears in *Computer-Aided Software Engineering*, (2nd Edition), E. Chikofski (ed.), IEEE Computer Society, 1993.

MS93
P. Mi and W. Scacchi. Articulation: An Integrated Approach to Diagnosis, Re-planning, and Re-scheduling. In *Proc. 8th. Knowledge-Based Software Engineering Conf.*, 77-85, Chicago, IL, 1993.

MS96
P. Mi and W. Scacchi. A Meta-Model for Formulating Knowledge-Based Models of Software Development. *Decision Support Systems*, 17(3):313-330. 1996.

Ni94 M. Nissen. Valuing IT through Virtual Process Measurement. *Proc. 15th. Intern. Conf. Information Systems*, Vancouver, Canada, 309-323. December 1994.

Ni96 M. Nissen. *Knowledge-Based Organizational Process Redesign: Using Process Flow Measures to Transform Procurement* unpublished Ph.D. Dissertation, IOM Dept., USC School of Business Administration, Los Angeles, CA, January 1996.

NS91 J. Noll and W. Scacchi. Integrating Diverse Information Repositories: A Distributed Hypertext Approach. *Computer*, 24(12):38-45, Dec. 1991.

O87 D. O'Leary. Validation of expert systems - with applications to auditing and accounting expert systems. *Decision Sciences*, 18:468-486, Summer 1987.

OG90
T. O'Leary, M. Goul, K.E. Moffit, and A.E. Radwan. Validating Expert Systems. *IEEE Expert*, 5(3):51-59, June 1990.

Sc89 W. Scacchi. The Power of Domain-Specific Hypertext Environments. *Jour. Amer. Soc. Information Science*, 40(3):45-53, 1989.

SM93
W. Scacchi and P. Mi. Modeling, Integrating, and Enacting Software Engineering Processes. In *Proc. 3rd. Irvine Software Symposium.* Irvine Research Unit in Software, University of California at Irvine, April 1993.

Sch94
R. Schank. Active Learning through Multimedia. *IEEE Multimedia* 1(1):69-78, Spring 1994.

ST96 P.G. Selfridge and L.G. Terveen. Knowledge Management Tools for Business Process Support and Reengineering. *Intelligent Systems in Accounting, Finance, and Management*, 5(1):15-24, 1996.

SG96
A. Sheth, D. Georgakopoulos, S. Joosten, M. Rusinkiewicz, W. Scacchi, J. Wileden, and A. Wolf. Report from the NSF Workshop on Workflow and Process Automation in Information Systems. Technical Report UGA-CS-TR-96-003, Dept. of Computer Science, University of Georgia, October 1996. http://lsdis.cs.uga.edu/activities/NSF-workflow/

SS72 R.F. Simmons and J. Slocum. Generating English Discourse from Semantic Networks. *Communications ACM*, 15:891-905, 1972.

SZ95 E.W. Stein and V. Zwass. Actualizing Organizational Memory with Information Systems. *Information Systems Research*, 6(2):85-117, 1995.

Su87 L.A. Suchman. *Plans and situated actions: The problem of human-machine communication* Cambridge University Press, New York, 1987.

Vo93 L. Votta. Comparing One Formal to One Informal Process Description. In *position paper circulated at the 8th. Intern. Soft. Process Work.* Dagstuhl, Germany, IEEE Computer Society, February 1993.

YM94

E.S.K. Yu and J. Mylopoulos. Understanding "why" in software process modeling, analysis, and design. *Proc. 16th. Intern. Conf. Software Engineering*, Sorrento, Italy, 159-168, 1994.

YM96

E.S.K. Yu and J. Mylopoulos. Using Goals, Rules and Methods to Support Reasoning in Business Process Reengineering.

YML96

E.S.K. Yu, J. Mylopoulos, and Y. Lesperance. AI Models for Business Process Reengineering. *IEEE Expert*, 11(4):16-23, August 1996.

Supporting Distributed Configuration Management in Virtual Enterprises

John Noll and Walt Scacchi

ATRIUM Laboratory
Marshall School of Business
University of Southern California
Los Angeles, CA 90089-1421

Abstract. This paper presents a semantic hypertext-based framework called DHT that supports distributed software configuration management, provides transparent access to heterogeneous, autonomous software repositories, and enables an implementation strategy with low cost and effort. We show how DHT solves the practical problems of sharing and updating heterogenous multi-version software in a virtual enterprise of distributed teams, integrating existing CM tools and environments, executing CM processes to coordinate development activities across wide-area networks. This is when the process model is represented as a user navigable hypertext graph whose nodes associate process steps, user roles, and associated tools with designated software product versions and configurations. Furthermore, we show that this can require the support for alternative policy models for the commitment of software updates into local CM repositories. Overall, these capabilities provide support for product-centered enactment of CM policies and processes across a virtual enterprise of teams connected via the Internet.

1 Introduction

Software development in the future will increasingly take place in settings where everything is potentially distributed. Software development teams will be highly decentralized, both physically and organizationally. Software will increasingly be produced by loosely coupled "virtual enterprises" composed of development teams from different organizations who collaborate on specific projects across a wide-area information infrastructure, then disband to form new alliances for other projects. Participants in these virtual enterprises will want to retain a high degree of autonomy over their own process activities, tools/environments, configuration management policies, and data. However, the dispersed developers will still need to create, access, or update software artifacts common to the development effort. These conditions give rise for the need new approaches for Configuration Management of shared software artifacts, repositories, and processes in virtual enterprises.

The following scenario illustrates: A loose collaboration among customer, consultant, vendor, and software contractor organizations is formed to enhance a legacy system. In this instance, the contractor is required to integrate the vendor's off-the-shelf user interface software package with the customer's legacy simulation system, according to the process specified by the consultant. Further, the contractor must access, reference, and link the customer's requirements specification to corresponding components in the integrated version of the legacy system, the vendor's user interface package, and the contractor added integration code, to support requirements traceability, as specified by the process model provided by the consultant. Each organization's team has its own tools, software artifacts, repositories (e.g, the customer has a SQL/RDBMS repository, the vendor has a conventional network file system as its repository, the contractor uses an RCS-based repository, and the consultant has a knowledge-based process library repository). Furthermore, each team's repositories may support different CM policy models (e.g., long transaction model for the customer's RDBMS repository, composition model for the vendor's NFS, check-in check-out model for the contractor's RCS repository, and change set model for the consultant's knowledge base repository) [10]. Each team also requires access to at least part of the others' artifacts. Also, each will need to update and expand the shared set of products that represents the project's collective output and deliverables.

The example above raises the main issue addressed by this paper: the virtual enterprise may adopt a configuration management policy and process that is not directly supported by any of the participating entity's repositories or environment. How can the chosen policy be supported?

In our view, the basis of the problem for distributed CM in virtual enterprises comprises the following aspects:

- provision of standard CM services – component identification, configuration structure definition, control over concurrent updates, team and process support, etc. [8] – in a manner compatible with the use of a distributed software infrastructure.
- provision of CM services across heterogeneous (and legacy) CM tools, data models, repositories, update control policy models [10], etc., while maintaining local/remote CM administrative autonomy and access control.
- provision of languages, notations, and/or executable environments for identifying and configuring heterogeneous software artifacts (atoms) and product structures (graph-structured aggregates) that relate software development products (software source code, development artifacts, executables, documentation), processes, organizations (location, roles, team members), and networked infrastructure (computing platforms, middleware, versioned software tools and environments, and the Internet/WWW).
- provision of support for CM *dynamism* in the form of evolving product structures (e.g., dynamic reconfiguration of software systems) and evolving development processes (in response to local contingencies and ongoing process improvements).

In this paper, we present an approach for addressing each of these. It utilizes an operational prototype environment that provides a multi-level integration capability that emphasizes a simple implementation scheme. The remainder of this paper is organized as follows: in the next section, we present an overview of our approach that is based on the use of a semantic distributed hypertext data model, client-broker-server architecture, and supporting environment. We refer to this capability simply as DHT [34]. Following this, we discuss the DHT approach to the multi-level integration of various CM tools, policy/workspace models, and repositories. Then, we present an approach to software/CM process modeling and enactment using DHT-based hypertext browsing. We conclude with a discussion of related research, and our contributions.

2 Overview of the DHT Approach

CM is a fundamental part of the software development life-cycle. Thus, both the virtual enterprise and the individual participants will almost certainly have some form of CM policy and mechanism in place.

For the virtual enterprise, the mechanism will be a combination of repository support provided by DHT, and policies implemented as process specifications. Similarly, each participating organization will also have its own policies, which may be supported to various degrees by CM mechanisms and process enactment.

However, it is unlikely that all of the participants will have the same CM policies or mechanisms, and it is possible that the virtual enterprise will adopt a policy that is not followed by any of the participants. This could be due, for example, to government regulations, customer requirements, or lack of available CM technologies.

Thus there are two issues that must be addressed in providing CM support for the virtual enterprise:

- The mechanisms and policies of participating organizations must remain intact for local use. This means that they cannot be modified to conform to either the hypertext view or the needs of the virtual enterprise.
- The software hypertext must be able to support whatever CM model is adopted by the virtual enterprise itself.

The first issue implies that the hypertext data model must be able to represent the concepts inherent in participating repositories, including modules, configurations, and version histories. The second issue means that it must be possible to overlay a global CM model onto the collection of objects and relationships exported from the participating repositories, and to transform objects in that model into objects conforming to the local model and policy.

At the same time, not all participating repositories have support for tracking, storing, and retrieving multi-version information objects, or configurations of objects. The Unix file-system, for example, does not have a built-in concept of versions; these are provided by higher level tools like RCS. RCS, in turn, does not support configurations as first class objects; these must be modeled using ad hoc techniques such as scripts and local conventions. Thus, the software hypertext data model's version and configuration concepts must be compatible with repositories that do not have these capabilities.

In the following sections, we present an overview of the DHT data model and architecture.

2.1 Data Model

The DHT data model consists of three primitives: *nodes*, that represent content objects such as software modules or documents; *links* that model relationships among nodes; and *contexts*, that enumerate sets of links to allow specification of composite objects, like directories or containers, as sub-graphs. In a semantic hypertext such as DHT, nodes, links, and contexts are all first class objects having types, attributes, and unique object identifiers (oids) associated with each. In addition, links have *anchors*, that specify regions or sub-components within node contents to which the endpoints of a link are attached.

Contexts enumerate, but do not actually contain, links. Thus, a link can be a member of several contexts, making it possible to construct different views of the same objects by imposing different structures as described by links among those objects. Contexts are also first class objects, and so may serve as the endpoints of links. Thus, contexts serve as a representational medium for specifying configurations of software objects, such as artifact and document aggregates, software product and process models, etc. as well as their interrelationships.

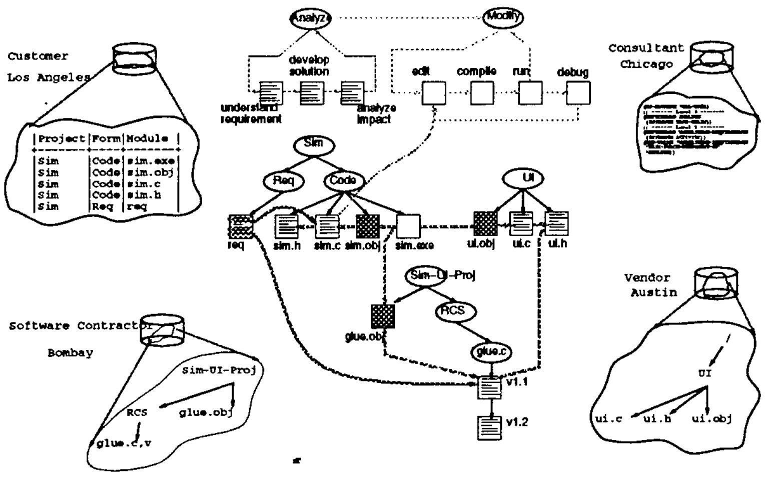
A fixed set of generic operations can be applied to DHT typed objects: *create, delete, read,* and *update* an object. The owners or administrators of given repository can elect to provide any subset of these operations (e.g., segmented by user groups, network location, or by type of client), as appropriate for the level of access they intend to offer. In addition, any operation can be performed by a single repository on its own objects. Cooperation among repositories is not required; thus, the model preserves a high degree of local repository autonomy.

2.2 Architecture

The DHT architecture is based on a client-broker-server model. Clients implement all application functionality that is not directly involved in storage management. Thus, a client is typically an individual tool (e.g., Emacs, gcc), but may be a complete software development environment.

Software artifacts are *exported* from their native repository server through object brokers called transformers. A *transformer* is a kind of mediator that exports local objects (artifacts and relationships) as DHT nodes and links, and translates DHT messages into local repository operations. Note that from the repository viewpoint, the transformer appears to be just a client-side tool/application.

A request-response style communication protocol implements the operations specified in the DHT data model, and includes provisions for locating servers and authenticating and encrypting messages. The protocol also provides a form of time-stamp concurrency control to prevent lost updates [23]. This provides a minimum concurrency control mechanism at the lowest integration level that is compatible with the widest possible variety of component repositories. More sophisticated concurrency control strategies are left to implementation at the client level through specific software process models (Section 3).



Project	Form	Module
Sim	Code	sim.exe
Sim	Code	sim.obj
Sim	Code	sim.c
Sim	Code	sim.h
Sim	Req	req

Fig. 1. DHT approach providing distributed CM in a virtual enterprise.

Figure 1 shows how a virtual enterprise repository might be constructed from component repositories using the DHT architecture and data model. A variety of data models and storage management mechanisms are translated into DHT hypertext primitives, forming a logically centralized, global hypertext transparently accessible to all participants. The details of each repository's access protocol and schema are hidden below this global integration layer.

Our experience to date has been that transformers for new repositories can be developed with modest effort (i.e., hours to days), based on reusable server templates that are augmented with code to interact with specific repositories.

Overall, the client-broker-server architecture that DHT employs provides the basis for addressing distributed CM in a virtual enterprise. Accordingly, we turn to examine in more detail why and how this architecture works by focusing on client-side tool-CM repository server integration mechanisms.

2.3 Example: RCS

The RCS transformer serves to demonstrate several DHT features. First, because RCS repositories (and repositories based on similar mechanisms) are so common in software development, it is important to be able to demonstrate that DHT is capable of faithfully representing their contents. Second, the RCS transformer presents some interesting challenges, in terms of modeling and implementation. Finally, it is an example of the added value a DHT transformer implementation can bring.

In the RCS system, the entire version history of a module is stored in a single file. Shell commands allow developers to retrieve specific versions by version number, create new versions, and examine the state of a module's version history. A numbering scheme identifies each version and its relation to other versions; the version history may contain branches and merges. RCS versions have attributes such as the version name, number, status, etc.

It is straightforward to translate these concepts into DHT primitives. Each module's version history is translated into a context. The version history itself is represented by links, while each specific version becomes a separate node, identified by its module name and version number. Attributes naturally become node attributes.

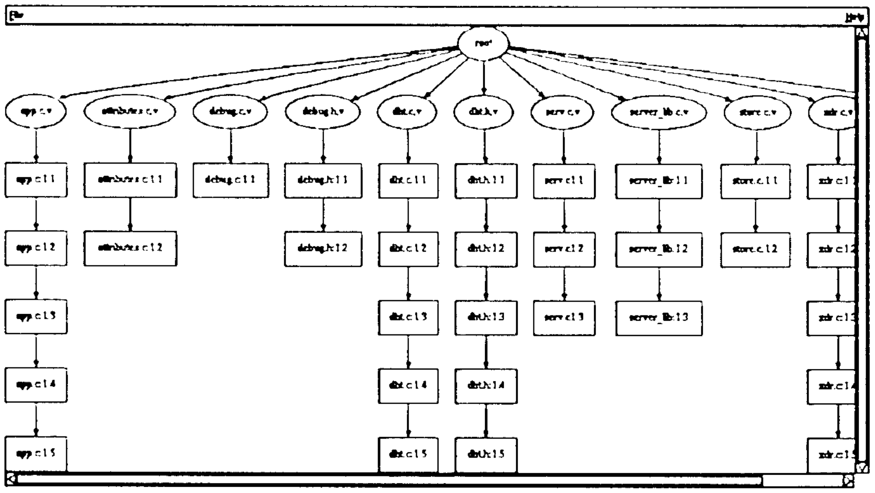


Fig. 2. DHT view of an RCS repository.

In addition, the entire collection of RCS version files is collected into a larger context, in order to capture the common notion of a collection of RCS files being associated with a project or subsystem. The resulting hypertext view is shown in Figure 2; this image shows how the DHT infrastructure library's RCS archive is exported.

2.4 SUPPORTING CONFIGURATION MANAGEMENT

Feiler [10] found four CM models in commercial software development environments:

1. The *check-out/check-in* model. This is the familiar model implemented by RCS.
2. The *composition* model. This model represents configurations as a set of *components* (equivalent to RCS modules) and version selection rules that specify which version of each component should be included in the configuration.
3. The *long transaction* model maintains versions of configurations rather than individual components. Developers copy configurations into private workspaces, where they modify individual components. The entire workspace is committed to create a new version of the configuration when the developer (or a verification board) is satisfied with the modifications.
4. The *change set* model focuses on the set of differences (the *delta*) between two versions of a component or configuration. In this model, deltas are first class entities that, when applied to a baseline object, produce a particular version of a configuration.

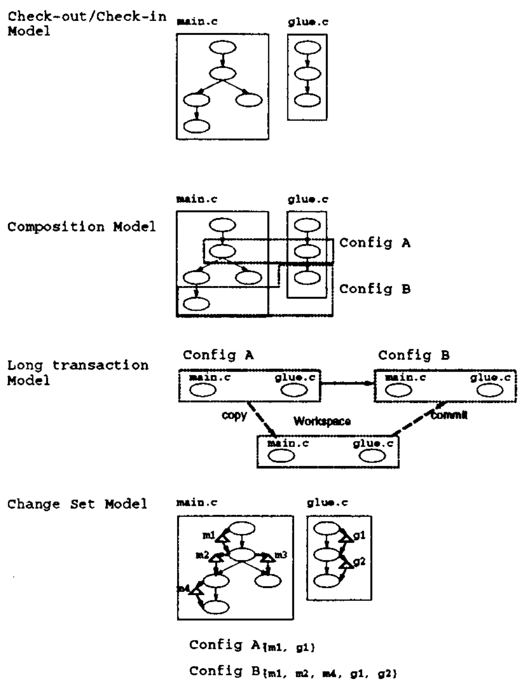


Fig. 3. Configuration Management Models.

Figure 3 shows how these models are described using the DHT data model. Specific versions of a module are translated into nodes with separate object identifiers. The relationships between successive versions, including branching and merges, are represented by static links. The entire collection of versions is exported as a context. Configurations consisting of a set of specific versions of each module making up a program or subsystem are also modeled by contexts.

Note that this approach is entirely compatible with repositories that do not provide versioning; in such cases, each module has a single version, and there are no history links or version contexts. Configurations can contain links to these single version modules and versioned nodes in the same context.

3 Incorporating Process Enactment

The CM models introduced in the previous section specify how components are organized into versions and configurations in a repository. They also dictate, at a high level, how components, versions, and configurations are created. However, the details of exactly how and when these events occur, and who may initiate them, are determined by the specific policies of each organization. So, for example, two organizations both using the check-out/check-in model may have different policies for when a module may be checked in to create a new version: one organization might allow the developer individual discretion, while the other may require the module to pass unit test and configuration review board approval before check-in to the repository.

Thus, CM policies determine specifically how the CM model is used in practice. These policies are part of the software development *process* followed by the development organization to produce software products.

A software process is a partially ordered set of tasks performed to develop software. A software process *model* is a description of a software process. If the description is sufficiently formalized, it is possible to execute process models for simulation, analysis, and *enactment*. Enactment, in turn, is a computer-supported cooperative activity involving one or more users. Users perform process steps using integrated tools to create, update, or otherwise manipulate designated software development products, according to the process fragments assigned to user roles.

Software process enactment uses the formal description of a software process to guide, monitor, and control the process by having a process interpreter or engine execute a formal process description.

Guidance involves leading developers through the process by issuing prompts or notifications as to what tasks should be performed at a given time.

Monitoring allows managers and engineers to assess the current state and progress of the process.

Control means ensuring the process is followed by restricting developer actions to those that conform to the process description.

A software process model has a natural representation as a hypertext graph. Nodes in the graph represent tasks or process steps, while links specify both the order in which the tasks should be performed and the products on which they should be performed. The resulting nodes, links, and contexts can be browsed and followed just like other hypertext graphs. A hypertext-based process model that links tasks to tools, user roles, and products thus provides a means both for sharing information and coordinating work in a virtual enterprise.

A DHT-based software process model contains three types of links:

1. *Process decomposition* links. These model the decomposition of high level tasks into lower level, smaller tasks, and eventually into primitive actions. These are static links that do not change unless the model itself is modified to correct errors or reflect changes in policy.
2. Task and action *precedence* links. These specify the order in which tasks should be performed. These are also static links that evolve slowly.
3. *Resource* links. These model the relationship between a particular product node and the tasks which should be performed on it at a given time, as specified by the process model.

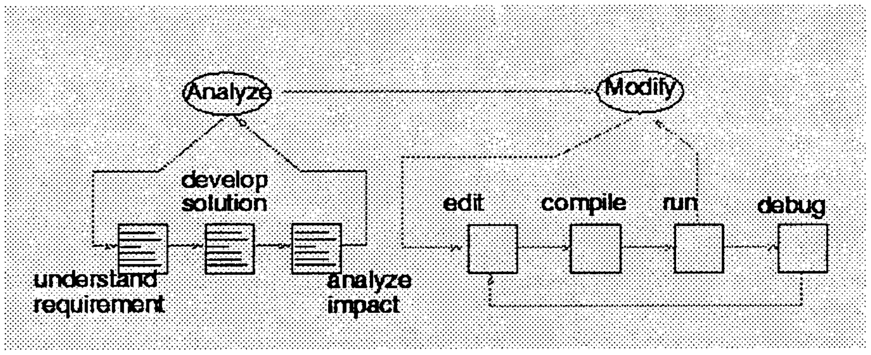


Fig. 4. A simple software process model showing decomposition and precedence links.

As an example, the following is an informal description of the process for modifying a module:

1. Retrieve module.
2. Edit module to implement changes.
3. Compile module.
4. Unit test module.
5. if the unit test is successful, create a new version of the module; otherwise,
6. Debug the module and return to step 1.

At a given time, many *instances* of the process may be active, on different modules by different developers. Each instance has separate process *state* [27, 18] including the module being modified, the developer doing the modification, the last step completed, etc. To support process enactment, it is necessary to keep track of this state for each process instance [18], in order to guide the developer through the process tasks in the appropriate sequence.

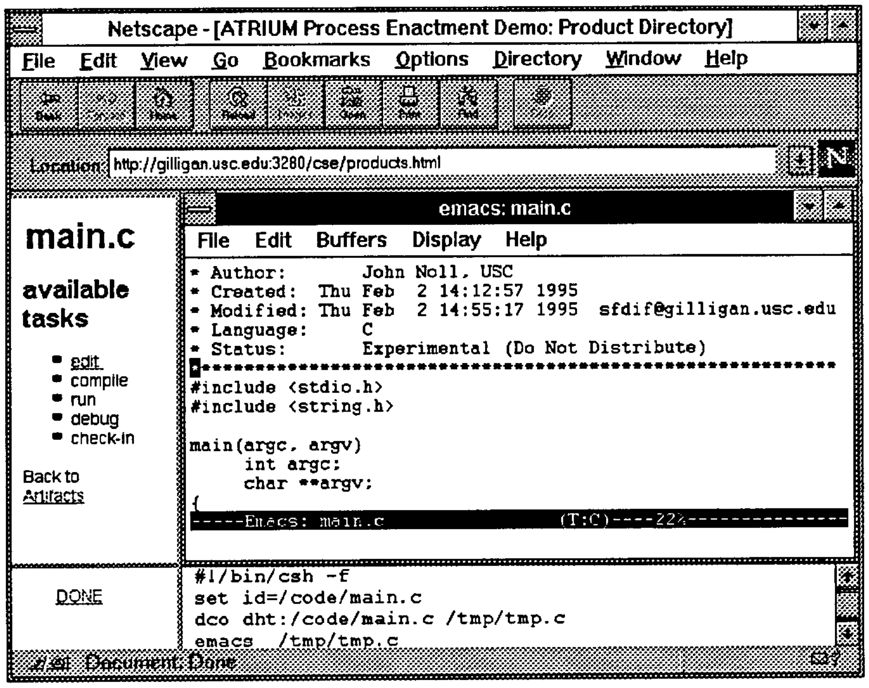
Process state is represented by Resource links. Each process instance has an associated context that contains the resource links that define the state of that instance. These links specify, for each process instance, which products to perform tasks on, what tools to use in performing the task, and the agent (developer) who is to perform the task. Also, a "current task" link points to the task currently being performed; this link serves the same purpose as a program counter, allowing the enactment mechanism to determine which task is pending for each process instance.

To enact a process in DHT, a user (developer, analyst, manager) visits the process instance context, then chooses the current task by following the current task link. This link resolves to a task node, which may be an executable script or a description of the task to be performed. In the former case, resource links resolve to the product nodes to which the script should be applied. In the latter case, the task description has resource links which the user can use to locate the resources and tools required to perform the task. In either case, a dynamic "task done" link causes the process instance state to be updated as a side effect of resolving the destination; it updates the current task link to point to the next task as determined by the task/process precedence links that are part of the process hypertext specification. Thus, process enactment is achieved by visiting nodes in the process graph, and following resource links in the context of the process instance.

For example, in the software process model in Figure 4, process task flow is indicated by a directed process graph. Suppose the user is to enact an instance of this process. Figure 5 displays a user interface view (using a common WWW hypertext browser) of this process. The developer is currently visiting the *main.c* product component, with the "edit" task pending (indicated by underscore and color). The user performing the edit task creates or updates *main.c* using the tool,"emacs".

In the screen view of Figure 5, the "edit" link marker indicated in the upper left frame of Figure 5 appears as it usually does for WWW browsing, whereas the lower right frame displays the contents of the link's destination (as task node) as a C-shell script. The user's selection of the "edit" task thus triggers execution of this script.

External events result in objects appearing in the virtual repository. This includes real creation of new objects, as well as initial export and discovery of legacy objects. In many cases such events should result in the instantiation of process instances. For example, when a new Engineering Change Notice (ECN) or Modification Request (MR) is released (created), a new instance of the modify process shown above should be created.


Netscape - [ATRIUM Process Enactment Demo: Product Directory]
File Edit View Go Bookmarks Options Directory Window Help
Location: http://gilligan.usc.edu:3280/cse/products.html
main.c
available tasks
edit
compile
run
debug
check-in
Back to Artifacts
DONE
emacs: main.c
File Edit Buffers Display Help
Document: Done

```
* Author:       John Noll, USC
* Created:  Thu Feb  2 14:12:57 1995
* Modified: Thu Feb  2 14:55:17 1995  sfdif@gilligan.usc.edu
* Language:     C
* Status:       Experimental (Do Not Distribute)
*******************************************************
#include <stdio.h>
#include <string.h>

main(argc, argv)
     int argc;
     char **argv;
{
-----Emacs: main.c                (T:C)----22%--------------
```

```
#!/bin/csh -f
set id=/code/main.c
dco dht:/code/main.c /tmp/tmp.c
emacs  /tmp/tmp.c
```

Fig. 5. A view of a DHT product-centered process UI.

Events fall into two categories:

- **Creation events**. These events occur when a new object is created, or a legacy object is first exported.
- **Change of state**. This occurs when an existing object is modified, resulting in a new value for one or more of its attributes.

In a centralized environment, the database or repository would detect both of these types of events, and implement process instantiation via triggers or event notification to clients.

In an autonomous, distributed environment, event capture is more problematic, since there is no single component through which events pass that can capture and handle them. Thus, event handling becomes a client responsibility: the client, upon discovering a new object or change of state in an existing object, takes some action such as instantiating a new process instance.

A potential problem arises when two clients detect the same event: should both handle the event? If not, how do they coordinate to decide which will act? In the case of process instantiation, several solutions are possible:

- Process instantiation must be idempotent, so events can be handled any number of times. This is difficult, since process threads by nature have state that changes once they are created.
- Clients must coordinate their activities, so that only one client handles each event. In a highly distributed environment, this may be impossible due to network partitions and race conditions.
- Processes themselves (the tasks and actions within a process) must be idempotent, so that multiple processes instantiated on the same event perform as if only one process was enacted. It is not clear that this is possible since by nature processes modify the state of the environment.
- Processes must be designed in such a manner that it makes sense for each client that *can* instantiate a process to do so. This is possible if process instances are bound both to the *products* they modify and the *agent* that performs them.

The last instance above is the most straightforward to implement, and seems to best match the nature of process enactment: agents assume roles to enact processes on products. Thus, it makes sense for a process instance to be bound to the products it modifies and the agent-role that performs it. Since clients also are associated with agents (users), the client can easily locate and examine state associated with the agent to determine when it is appropriate to handle an event.

Attribute	Value
source	`{[type $node]==DHT:ECN && [status $node]=="released"}`
source_anchor	Global
dest	`{eval [get-contents process:modify-constructor]}`
dest_anchor	Global
type	DHT:Process-constructor

Fig. 6. An example link specification for instantiating the modify process.

DHT dynamic links can implement this behavior, and thus provides a basis for supporting CM dynamism. This is achieved in the following manner: the source predicate examines the state of a given object; the client evaluates the predicate on objects as they are discovered or visited. The resolution function examines a state object associated with the agent to see if a process instance context already exists that satisfies the source predicate; if not, it creates a new context for the process described by the destination of the link. Thus, the state object associated with an agent is a context containing links to the agent's active process instances.

Note that, because agents are the most common initiators of events, detection of events by dynamic links associated with a particular agent can be nearly instantaneous.

An example instantiation link is shown in Figure 6. The source predicate is true when applied to released ECN nodes. The resolution function retrieves the contents of a "constructor node" that specifies a script for creating new instances of the process. This script would examine the users state context to see if an instance of the process is already linked to the ECN node; if not, it creates a new process instance context and adds a link to the agent state context.

The significance of this approach to modeling process instances is that *the mechanism for process enactment is embedded entirely within the process representation*, as the source predicates and resolution functions of available-task links. In contrast to other enactment systems which employ an environment accessed through a process-based user interface [27, 16], process-aware tools [46], or process state database [18] to execute process specifications, DHT process models can be enacted simply by browsing the process hypertext using any DHT browser or tool. This means that support for wide-area process enactment can be introduced into an existing environment with minimal disruption.

4 Related Work

We have approached integration by providing the illusion of a central repository through the introduction of a layer between storage managers and users of data. Such a layer provides a logically integrated virtual repository of data objects that conform to a hypertext data model. The DHT architecture provides the physical integration of participating repositories necessary for users to access instances of data objects.

Other research uses the same general approach we have taken, but with different data models and results. For example, network file-system solutions [22, 40, 42] implement a common global file-system at the integration layer, where each repository exports its objects as files in a single unified directory tree. However, network file-systems are a lowest-common denominator solution [44]: the directory-file model lacks explicit constructs to represent the numerous relationships that exist among software artifacts [37]. As a result, ad-hoc techniques such as file naming conventions, and numerous tool-specific databases like Makefiles, tag files, etc. are required to augment the basic directory-file model. Consequently, information sharing is at a rudimentary level.

The multi-database approach [4] occupies the other extreme: here the integration layer provides a relational or object-oriented data model with explicit relationship types. This would seem to solve the relationship problem of the network file-system; both of these models have been used in conventional environments. However, the complexity of constructing and maintaining a single global schema that captures all of the concepts present in each participating repository, combined with the requirement that integrated repositories have DBMS functionality (query language, data schema, transactions, etc.), generally makes this approach costly and difficult to implement [9]. CORBA-based implementations such as NUCM [19] face similar problems. Note, however, that CORBA could serve as the foundation layer on which the DHT protocol could be implemented; we chose a different approach largely because of the relative immaturity of CORBA implementations at the time.

A number of research projects have applied hypertext to software object management, including the Hypertext Abstract Machine (HAM) [6], the Documents Integration Facility (DIF) [13, 14], and HyperCASE [7]; however, these are based on a single, centralized repository architecture. Notable exceptions are PROXHY [20], Chimera [2], and Endeavors [5]. In contrast to DHT, PROHXY and Chimera focus on adding new links among existing artifacts. However, existing relationships among the artifacts or data repositories are not translated into links. ISHYS [15] and sigmaTrellis [45] explored software process modeling and enactment via hypertext, although in a centralized architecture. More recently, HOSS [35] and Endeavors [5] have addressed process support, and have extended their hypertext mechanisms support distributed capabilities similar to DHT, but with some differences. For example, HOSS, Endeavors, and DHT all provide a semantic hypertext modeling and process enactment capability, but differ in how process behavior is specified (as methods attached to activity objects versus as predicates attached to dynamically updated precedence links). Also, Endeavors and HOSS are addressing issues of versioning, but based on the available reports, have not yet addressed the range of standard CM services [8] and CM update policy models [10] that have been addressed by DHT and NUCM [19], nor the multi-tactic tool integration schemes of DHT or DESERT [41].

Finally, recent surveys of conventional and process-centered software engineering environments [21, 12], underscore the growing trend toward the support of process life cycle activities, such as modeling, enactment, and repository management [25]. Many such environments rely upon the use of a centralized object management system that may be implemented using a relational DBMS-controlled object server or object-oriented DBMS as its repository, with static graph-based or reactive rule-based process modeling and enactment capabilities (e.g., Marvel [3], SMART [16], Articulator [28], and dozens of others in the U.S., Europe, and Japan [12]). However, the reliance on a DBMS for object management services or CM implies the need for a transaction manager. In contrast, software hypertext environments such as DHT, HOSS or Endeavors do not assume nor require a central database transaction manager, yet provide process modeling and enactment, as well as support for heterogeneous object management systems, all within a wide-area information infrastructure. DHT can incorporate both DBMS and non-DBMS repositories, whether specific to CM or not. In addition, DHT supports navigational browsing and versioning, multiple workspace models, and is compatible with existing Internet resource discovery mechanisms [36].

Thus, in our view, we see that software hypertext browsing, linking, and process enacting infrastructures seem to provide a promising strategy for addressing the problem of distributed CM for virtual enterprises.

5 Conclusion

We began with the assertion that software development in the future will be performed by cooperating development teams. These teams will be autonomous, widely distributed, work concurrently, and loosely associated in virtual enterprises. However, they will need to create, share, and concurrently update software artifacts, as well as coordinate and manage the relationships among them, that form the products of the development process.

We observed that the conventional software environment architecture is inadequate to support software development and distributed CM within virtual enterprises, because it relies on a central repository to serve as the medium of data sharing among users. Thus, the research problem we addressed was how can virtual enterprises share, manipulate, and update data among their loosely-coupled but process-coordinated participants, without a single physically centralized CM repository nor update policy model.

We proposed a solution based on providing an integration layer between autonomous software repositories servers and their clients, that would provide the appearance of a central repository while maintaining the distributed physical environment. DHT achieves three levels of integration: *logical* integration by describing data and operations on them in terms of a common hypertext data model; *physical* integration via a distributed architecture for access to autonomous repositories; and *process integration* for definition and binding developer roles to process tasks to client-side tools to product components provided by the physical repositories according to the logical scheme of the virtual enterprise.

The DHT approach has the following benefits:

- **Evolutionary approach to integration and CM:** It is possible to migrate a conventional Unix toolbox-oriented environment to DHT without recompiling any of the individual tools. This is because DHT offers several levels of tool integration, depending on the degree of "hypertext awareness" desired for a given tool. This incremental evolutionary approach to incorporating existing tools into a DHT environment enables tool integrators to make tradeoffs between integration effort and hypertext functionality. This gives administrators great flexibility to preserve existing investment in tools and training while simultaneously obtaining the advantage of DHT's integration and hypertext capabilities.

- **Transparent process enactment:** By representing software processes as hypertext graphs, DHT achieves enactment without the need for an explicit process interpreter or environment. This is a significant advantage in a virtual enterprise, where each participant may already have a favorite development environment in-place. The DHT approach allows process enactment to co-exist with existing tools and environments. Thus, DHT provides support for integrating products, organizations (and developer roles, not shown), tools, and processes that may all be distributed across the Internet in a transparent manner.
- **Comprehensive solution:** The most significant contribution of DHT is the way it applies the features of hypertext to data integration, combining intrinsic support for user interaction with data modeling and access. DHT combines the advanced data modeling capabilities of semantic nets with the natural navigation-based access of file systems, and the intuitive direct-manipulation browsing features of hypertext. This means that as soon as a transformer is put in place to export data from a particular CM repository, the user interface and access operations to those data are also in place. Furthermore, both the user and access interface can conform to a single global CM policy model which transcends local organizational policy models, therefore maintaining highly transparent access to heterogeneous data. The result is effective yet low in implementation cost.

We set out to develop a solution to the problem of providing for the sharing and distributed configuration management in a virtual enterprise of autonomous development teams using: semantic hypertext data modeling and management appropriate for software artifacts and relationships; transparent access to heterogeneous, autonomous legacy repositories; multi-level tool integration; process modeling and wide-area enactment; and low implementation cost. In so doing, DHT solves practical problems of sharing data, coordinating work processes, and supporting distributed CM services in a virtual enterprise, and establishes a basis for continuing research in integrating heterogeneous software object management repositories, data models, and implementation architectures using easily navigated wide-area hypertexts.

6 ACKNOWLEDGMENTS

This work is sponsored by the Office of Naval Research under grant N00014-94-1-0889. No endorsement implied.

References

1. Evan W. Adams, Masahiro Honda, and Terrence C. Miller. Object management in a CASE environment. In *Proc. 11th Intl. Conf. on Software Engineering*. IEEE and ACM, 1989.

2. Kenneth M. Anderson, Richard N. Taylor, and E. James Whitehead, Jr., Chimera: Hypertext for heterogeneous software environments. In *European Conf. on Hypermedia Technology*, Edinburgh, Scotland, September 1994.
3. Israel Ben-Shaul, Gail Kaiser, and George Heineman, An Architecture for Multi-User Software Development Environments. In *Proc. 5th. ACM SIGSOFT/SIGPLAN Symposium on Practical Development Environments*, 1992.
4. M. W. Bright, A. R. Hurson, and Simin H. Pakzad. A taxonomy and current issues in multidatabase systems. *IEEE Computer*, March 1992.
5. G.A. Bolcer and R. Taylor. Endeavors: A Process System Integration Infrastructure. *Proc. Intl. Software Process Conf.* (to appear), December 1996.
6. Brad Campbell and Joseph M. Goodman. HAM: A general purpose hypertext abstract machine. *Communications of the ACM*, 31(7), July 1988.
7. Jacob L. Cybulski and Karl Reed. A hypertext based software-engineering environment. *IEEE Software*, March 1992.
8. S. Dart. Concepts in Configuration Management Systems. In *Proc. Third Intern. Workshop on Software Configuration Management*, ACM SIGSOFT, 1-18, 1991.
9. D. Fang, J. Hammer, D. McLeod, and A. Si. Remote-exchange: An approach to controlled sharing among autonomous, heterogenous database systems. In *Proc. IEEE Compcon*, San Francisco. IEEE, February 1991.
10. Peter H. Feiler. *Configuration Management Models in Commercial Environments.* Technical Report CMU/SEI-91-TR-7, Software Engineering Institute, Carnegie Mellon University, March 1991.
11. S. Finger, M. Terk, E. Subrahmanian, C. Kasabach, F. Prinz, D.P. Siewiorek, A. Smailagic, J. Stivoric, and L. Weiss, Rapid Design and Manufacture of Wearable Computers, *Communications of the ACM*, 39(2):63–70, 1996.
12. Pankaj K. Garg and Mehdi Jayerzi (eds.), *Process-Centered Software Engineering Environments*, IEEE Computer Society, Los Alamitos, CA, 1996.
13. Pankaj K. Garg and Walt Scacchi. A Hypertext Environment for Managing Configured Software Descriptions, *Proc. First Intl. Workshop Version and Configuration Control*, pp. 326-343, B.G. Teubner, Stuttgart, FRG, (January 1988).
14. Pankaj K. Garg and Walt Scacchi. A hypertext system for software life cycle documents. *IEEE Software*, 7(3):90–99, May 1990.
15. Pankaj K. Garg and Walt Scacchi. ISHYS: Designing an intelligent software hypertext system. *IEEE Expert*, 4(3):52–63, Fall 1989.
16. P.K. Garg, P. Mi, T. Phan, W. Scacchi, and G. Thunquest. The SMART Approach to Software Process Engineering, *Proc. 16th. Intnl. Conf. Software Engineering*, Sorrento, Italy, IEEE Computer Society, 341-350. 1994.
17. M. Hardwick, D.L. Spooner, T. Rando, and K.C. Morris, Sharing Manufacturing Information in Virtual Enterprises, *Communications of the ACM*, 39(2):46–54, 1996.
18. Dennis Heimbigner. The ProcessWall: A process state server approach to process programming. In *Proc. Fifth SIGSOFT Symposium on Software Development Environments*, Tyson's Corner, Virginia, December 1992.
19. A. van der Hoek, D. Heimbigner, and A. Wolf. A Generic, Peer-to-Peer Repository for Distributed Configuration Management. In *Proc. 18th. Intl. Conf. Software Engineering*, IEEE Computer Society, Berlin, 308-317, March 1996.
20. Charles J. Kacmar and John J. Leggett. PROXHY: A process-oriented extensible hypertext architecture. *ACM Transactions on Information Systems*, 9(4):399–420, October 1991.

21. Anthony S. Karrer and Walt Scacchi, Meta-Environments for Software Production, *Advances in Software Engineering and Knowledge Engineering*, Vol. 4, D. Hurley (ed.), World Scientific Press, 1995.

22. James Kistler and Mahadev Satyanarayanan. Disconnected operation in the coda file system. *ACM Transactions on Computer Systems*, 10(1):3–20, February 1992.

23. Henry F. Korth and Abraham Silbershatz. *Database System Concepts*. McGraw-Hill, 1986.

24. Y-J. Lin and S. Reiss. Configuration Management with Logical Structures. In *Proc. 18th. Intern. Conf. Software Engineering*, IEEE Computer Society, Berlin, 298–307, March 1996.

25. P. Mi, M-J. Lee, and W. Scacchi. A knowledge-based software process library for process-driven software development. *Proc. 7th. Knowledge-Based Software Engineering Conf.* Washington, DC, IEEE Computer Society, 121-132, 1992.

26. P. Mi and W. Scacchi, A knowledge-based environment for modeling and simulating software engineering processes, *IEEE Trans. Knowledge and Data Engineering*, 2(3):283-294, 1990.

27. P. Mi and W. Scacchi. Process integration in CASE environments. *IEEE Software*, 9(2):45–54, March 1992.

28. P. Mi and W. Scacchi. A knowledge-based meta-model for formulating models of software development processes, *Decision Support Systems*, (to appear), 1996.

29. National Industrial Information Information Protocol Consortium (Vision, Goals and Objectives Pages). See http://www.niiip.org.

30. B.A. Nejmeh. Internet: A Strategic Tools for the Software Enterprise, *Communications of the ACM*, 37(11):23–27, November 1994.

31. John Noll. *Software Object Management in Heterogeneous, Autonomous Environments: A Hypertext Approach.* PhD Dissertation, University of Southern California, 1997.

32. John Noll and Walt Scacchi. Integrating diverse information repositories: A distributed hypertext approach. *IEEE Computer*, 24(12):38–45, December 1991.

33. John Noll and Walt Scacchi. A hypertext system for integrating heterogeneous, autonomous software repositories. In *Proc. Fourth Irvine Software Symposium*, pages 49–59, Irvine, CA, April 1994.

34. John Noll and Walt Scacchi. Repository Support for Virtual Software Enterprise. In *Proc. California Software Symposium*, UCI-USC, Los Angeles, CA, April 1996.

35. P.J. Nurnberg, J.J. Leggett, E.R. Schneider, and J.L. Schnase. Hypermedia Operating Systems: A New Paradigm for Computing, *Proc. Hypertext '96*, ACM, Washington, DC, March 1996.

36. Katia Obraczka, Peter Danzig, and Shih-Hao Li, Internet resouce discovery services, *Computer*, 26(9):8–22, 1993.

37. Maria H. Penedo, Erhard Ploedereder, and Ian Thomas. Object management issues for software engineering environments; workshop report. In *SIGSOFT '88, Boston*, November 1988.

38. P. J. Plauger. *The Standard C Library*. Prentice Hall, 1992.

39. Jurgen Reuter, Stefan U. Hngen, James J. Hunt, and Walter F. Tichy. Distributed Revision Control Via the World Wide Web, In *Proc. Sixth Intl. Workshop on Software Configuration Management*, Berlin, Germany, March, 1996

40. Herman C. Rao and Larry L. Peterson. Accessing files in an internet: the jade file system. *IEEE Transactions on Software Engineering*, 19(6):613–625, June 1993.

41. S. Reiss. Simplifying Data Integration: The Design of the Desert Software Development Environment. In *Proc. 18th. Intern. Conf. Software Engineering*, IEEE Computer Society, Berlin, 398–407, March 1996.
42. Mahadev Satyanarayanan. The influence of scale on distributed file system design. *IEEE Transactions on Software Engineering*, 18(1):1–9, January 1992.
43. Walt Scacchi. A software infrastructure for a distributed system factory, *IEE/BCS Software Engineering Journal*, 6(5):355–369, 1991.
44. Peter Scheurmann, Clement Yu, Ahmed Elmagarmid, Hector Garcia-Molina, Frank Manola, Dennis McLeod, Arnon Rosenthal, and Marjorie Templeton. Report on the workshop on heterogeneous database systems. *SIGMOD Record*, 19(4), December 1990.
45. P. David Stotts. sigmaTrellis: Process models as multi-reader collaborative hyperdocuments. In *Proc. Ninth Intl. Software Process Workshop*, Airlie, Virginia, October 1994.
46. Richard Taylor, Frank Belz, Lori Clarke, Leon Osterweil, Richard Selby, Jack Wileden, Alexander Wolf, and Michal Young. Foundations for the Arcadia environment architecture. In *SIGSOFT '88, Boston*, November 1988.
47. Working Group on Versioning and Configuration Management of World Wide Web Content. http://www.ics.uci.edu/ ejw/versioning/

A Unix File-system Transformer

Since so many software objects are in files and file systems, it is important to provide transparent hypertext access to these objects. It is also useful to provide access to remote files that may not be conveniently mounted using the usual network file-system (NFS) mounting mechanism. Finally, there are implicit relationships within file-systems that can be made explicit by exporting them as hypertext links. As an example of the relative simplicity of transformer construction, as well as how local concepts are mapped onto the DHT data model, in this section we discuss the implementation of a file-system transformer.

The file system transformer accomplishes these goals by translating plain files into content nodes, and directories into contexts. The containment relationship between a directory and its contents is represented by hypertext links.

Implementing a file system transformer is straightforward. The file-system provides a number of system calls for accessing directories, files, and their attributes. As an example, we will examine the implementation of the get-object operation for the three types of file-system objects: directories, plain files, and links between a directory and its children.

Plain files Because of the similarity between hypertext nodes and files, the mapping from a file to a node is straightforward. The following pseudo-code illustrates:

```
1. Execute the stat() system call on the file specified by the
object identifier.  Convert the returned values into a list of
attribute name-value pairs.
2. Read the file's contents and append to the attribute list.
Return the resulting list as the reply to the request.
```

Directories Directories are slightly more complicated. A directory is translated into a context that specifies the parent-child links implied by the relationship between the directory and its children. Therefore, there has to be some way to encode these relationships into link ids for later retrieval.

The contents of a directory can be obtained using the *scandir()* library function. This returns a list of names of the files contained in the directory. These names are then used to construct link identifiers for the directory's links by concatenating the directory's path with the file name.

The resulting list of links is added as the contents attribute to the list of attributes obtained from the *stat()* system call, and returned as the reply.

Directory-child links The link identifiers returned in a directory context encode information on how to retrieve the data associated with the link; namely, the source directory path and destination file name. Given this information, it is possible to construct the link without actually accessing the file-system; however, to ensure accuracy the file system transformer executes *stat()* on each endpoint to verify that the link is still valid.

(Re)Engineering Research Grants Management: From Acquisition Reform to Knowledge Brokering at ONR

Walt Scacchi[1], John Noll[1], Cedric Knight[2], and Capt. Felton "Jay" Miller[3]

[1]ATRIUM Laboratory, University of Southern California, Los Angeles, CA.

[2]New Directions Technologies Inc., Ridgecrest, CA.

[3]Acquisition Directorate, Office of Naval Research, Arlington, VA

Email: Scacchi@gilligan.usc.edu

Abstract

In this paper, we briefly describe our approach and experience in a research effort focused on (re)engineering the activity of research grants management at the Office of Naval Research. We found that we could contribute to a substantial reduction in process cycle time and operational costs associated with the funding of thousands of research grant procurement actions. Accordingly, we focus our discussion on topics that underlie these results. We also observe that *knowledge brokering* is an area where a new R&D initiative could lead to more effective and efficient research funding and research program management, as well as serve the mutual self-interests of the Federal research funding agency and researcher communities.

Keywords:

Intelligent information integration, electronic transactions and electronic commerce technologies, knowledge-based process engineering, process-driven intranets for research management.

Introduction

We have been involved in a multi-year research project investigating ways to reinvent and reengineer corporate financial operations in military procurement and acquisition organizations. Most recently, this effort has been directed at the management of research grants by the Office of Naval Research using electronic commerce and knowledge-based process engineering technologies.

Through our research effort, we have found that dramatic reduction in the cycle time and operational costs of research grants management processes can be achieved. Internal ONR performance measures now reveal that a factor of 10 in reduction of process cycle time has been realized in procurement action lead time (PALT). PALTs, used for cross-industry benchmarking purposes, indicate the elapsed time at ONR from a research grant funding authorization to the time when the grant recipient can begin to expend their funding award. As more than 5000 research grant funding actions are performed per year at ONR (1995-1997), a reduction of average PALT from 70 days (in 1994) to 7 days (in 1997, with further reductions possible) represents a significant demonstration of acquisition reform at ONR. In addition, some of the information processing and workflow redesign that we collectively developed with ONR personnel has led to the identification of annual operational savings estimated in the range of $10M-$15M, and elimination of these expenses henceforth.

How did we achieve these results? What research methods and prototype tools did we employ to achieve these results? Can similar order of magnitude improvements be made in managing and "brokering" scientific research programs by ONR program managers and scientific officiers (i.e., the people who solicit and review research proposals, and collectively recommend funding actions)?

This paper seeks to briefly explain what we did in our research to achieve our results, and our thoughts for the extension of this line of research to other federal research grants agencies. Additional details beyond the scope of this paper can be found in presentation materials that have been posted on the WWW, or as listed in this paper's References. Furthermore, we also identify an emerging opportunity for further research, namely, how this effort can be expanded to address knowledge brokering processes at ONR and elsewhere. But first, we want to specifically address topics that are relevant to the Workshop on R&D Opportunities in Federal Information Services.

Knowledge-Based Process Architecture

Research projects at the USC ATRIUM Laboratory have primarily focused on the knowledge-based engineering of complex organizational processes. Since 1990, these efforts have investigated the development of tools, techniques, and concepts for engineering organizational *process architectures* in domains such as large-scale software engineering, new product development, supply chain logistics, corporate financial operations, software acquisition, and military procurement. These process architectures are knowledge representations that model the processes, products, organizational roles and team composition, information infrastructure, and development tools central to an organization in its "routine" work operations. A descriptive characterization of the knowledge ontology that we employ that allows us to rapidly transition our efforts across different organizational domains can be found elsewhere [MS96]. Similarly, the knowledge-based tools and techniques we employ in engineering these process architectures across the *process life cycle* is also described elsewhere [SM96]. Accordingly, in this study, our focus was directed at four major processes of ONR's research grants management activity: Grants Pre-Award (proposal solicitation), Award (funding decision and obligation), Administration (funds disbursement and field office operations), and Close-out (completion and reporting compliance).

Application Domain for Multiple Federal Agencies

In 1994, ONR awarded a research grant to the ATRIUM Laboratory to investigate the development of alternative process architectures for military procurement at the Naval Air Warfare Center, Weapons Division, in China Lake, CA. This effort was later extended to investigate ONR research grants management processes, principally those involved in ONR's Acquisition Directorate. In both situations, the ATRIUM Laboratory was configured to operate as a kind of *process reinvention collaboratory* [cf. KMW96] where personnel from NAWC/ONR could meet off-/on-site with USC researchers, supported with process elicitation, visualization, integration, and execution support tools that could be accessed over the Internet [NS91,SM96,NS97b].While our efforts were of modest scale, we note that

NAWC-WD is among the largest of the US Navy's 1000+ procurement centers, and ONR is the largest of DoD's research grant agencies (e.g., Darpa, AFOSR, ARO) in terms of grant actions, as well as one the Federal government's largest research grants agencies (NSF, NASA, DOE, etc.). Thus, results we might acheive through our research efforts may be applicable to a large number of government information service centers.

Effort Needed to Obtain Useful Research Results

The application domain of procurement and acquisition is not "rocket science." Instead, it is usually considered a back office business activity concerned with corporate financial operations, expenditure management, and status reporting. It is an activity that is governed by a large number of changing acquisition regulations and policies at Federal, DoD, and Navy levels. Federal Acquisition Regulations (FARs) apply to all government agencies, including those involved in funding research grants. In many ways, the processes and artifacts used to manage procurement contracts are quite similar to those used to manage research grants. Similarly, the personnel at ONR who administer grant awards from ONR's five field offices in the US, are also the same people who manage contract fulfillment obligations as well as FAR compliance and reporting requirements from research institutions. Thus, the domain of procurement and acquisition of research grants and service contracts for Federal agencies is likely to be highly tractable and sufficiently structured to enable successful domain knowledge engineering. Furthermore, what we learn about ONR's research grants management process architecture may be refined and tuned for application in other DoD or Federal research grants agencies with affordable effort.

Addressing Barriers to Resistance

Procurement and Acquisition Divisions are populated with personnel that are usually not specialists with advanced information technology. These people are not computer scientists, nor can they be expected to be familiar with knowledge-based tools, techniques, or concepts know being employed within the research community. Furthermore, they may often be expected to "resist" the intervention by "outsiders" whose purpose may be perceived as eliminating their jobs or administrative authority. Nonetheless, ONR like other government agencies, is under substantial pressure to accept increasing workloads with shrinking budgets.

Our approach to understanding the process architecture of research grants management activities at ONR was based on involvement, participation, and (intellectual) engagement of personnel from the top to the bottom of the organization chart. We needed to educate ONR personnel in our motives and methods, and they needed to educate us on the generic and circumstantial variants of their work processes and information flow. Two or three iterations were typically performed, particularly with key domain experts. Follow-up validations by other personnel not necessarily involved in these iterations were also performed. Furthermore, agreements were established early on between the research team and ONR personnel covering the following items:

- the research team would identify multiple opportunities for (re)design of work processes [Ni94,Ni96], information flow, and information integration (See Appendix for examples);
- effort would be directed at improving personnel effectiveness and workflow without increasing anyone's workload--personnel had to be more satisfied with the new work arrangements;
- the processes examined would be developed in three forms: AS-IS (present form), TO-BE (alternative process architecture), and transitional forms (steps taken in 30 day increments to

evolve from the AS-IS to TO-BE forms);

- no new personnel positions would be created;
- ONR personnel would make final decisions on the selection of improvement alternatives that would be implemented;
- any improvements to be implemented had to be "self-motivating" or enable local organizational incentives to increase the likelihood of their successful implementation and routinization.

As a result of these efforts and agreements, we found little or no resistance, since our efforts were defined and structured as inherently collaborative in purpose, method, and outcome.

Key Enablers and Support Technologies

Our method and agreement for research engagement as noted was a key enabler for achieving the results we did. Similarly, the knowledge-based process engineering tools, techniques, and concepts that we have been developing and experimenting with at USC for the past seven of so years, were a key enabler. This of course should be no surprise--we built them, we use them, we evolve them to meet our emerging research needs. We have an investment in making them a key enabler, as well as serving to differentiate our effort when competing for external research funding. Nonetheless, we cannot be satisfied on this basis alone. Instead, we must find ways to make our research technologies accessible as prototypes to external customers and users, such as personnel at ONR Headquarters and at its five field offices across the US. As such, part of our research effort has been directed at prototyping an Internet-based information infrastructure that could be used to capture, analyze, convey, prototype, demonstrate, and refine alternative architectures for organizational processes, such as ONR's research grants management activity [NS97b,NS97a]. Such an infrastructure must eventually be able to support activities associated with:

- identification of Fleet and basic scientific needs, leading to the establishment of new research programs
- preparation, review, revision, and distribution of electronic research proposal solicitations
- receipt and integration of external research grant awards or programs from other Federal agencies (Darpa, Nasa, DOT, etc.)
- preparation, submission, review and revision of electronic research proposals and budgets
- integration of multiple heterogeneous information systems and data repositories (INRIS, CAMIS, STARS, etc.) that at present asynchronously record research funds expenditures
- preparation, distribution, and administation of electronic research grant award packages, containing records of all grant actions pertaining to a research grant award (funding increase or decrease, incremental funding, grant renewals, no-cost funds extension, address changes, etc.)
- electronic data interchange for electronic invoicing and electronic funds transfer transactions between ONR and research institutions
- on-demand tracking and reporting on the status of in-progress procurement actions, and research program funding obligations, encumberances, and actual expenditures (expenditure management)
- field office monitoring of regulatory compliance, record keeping practices, and resource control systems (e.g., for tracking equipment or property purchased with research grant funds) at grant receiving institutions
- receiving, tracking, archiving, querying, retrieving, and browsing findings, reports, or online prototypes resulting from research grant awards
- conveying research progress and orchestrating advanced technology demonstrations for customers within the Fleet in order to substantiate, expand, or decrease further research program investments.

In turn, such a prototype can serve as testbed or process-driven intranet for ONR research management. Accordingly, we can employ this testbed to demonstrate delivery of a research project's inputs and outputs, in a form that can be accessed, engaged, and served across ONR's multiple sites. Otherwise, those familiar with the Electronic Research Administration initiative (NewERA) in which five Federal research agencies (ONR, NIH, DOE, AFOSR, DOT) currently participate may observe that such an infrastructure addresses a spectrum of NewERA concerns. However, in our research project, we have the constraint (or "luxury") of limiting our investigation to a single research agency (ONR), together with (a) the ability to identify and prototype alternative process architectures for research grants management, and (b) access to an (in-progress) integrated information infrastructure that can directly support activities such as those just noted.

Are the Research Results Rapidly Deployable and Demonstrable?

The relative ease with which our research results can be deployed outside of ONR is a matter of opinion. Nonetheless, we can demonstrate and provide WWW-based presentations on what we have done, how it was accomplished, and how it might be applied, reproduced, or reused in other Federal research grants agencies. To this end, we have developed an exploratory scenario for how external government agencies that fund research grants through ONR (e.g., Darpa, NASA) might operate. Specifically, a recurring problem of internal and external research programs is tracking the status of *expenditure management*: how much unallocated research funding is available at present for funding obligations, and how much is being or has been actually spent (encumbered or expended). For example, if PALTs require process completion times measured in months or weeks, then uncertainty, misunderstanding, and organizational inefficiency can occur relative to the status of funds availability. Reducing PALTs to days (or even to hours!) can reduce some of these dilemmas. Subsequently, being able to address practical problems such as expenditure management, which turn out to be of great importance in research funding and reallocation decision-making (i.e., "strategic" decision-making situations by research program officiers and division managers), may likely determine the eventual success in being able to apply research efforts such as ours in other settings. Such a scenario can be described in more detail at the Workshop, if the opportunity arises.

Related Areas for Research Attention: Knowledge Brokering

Most of what has been described so far focuses on acquisition and research grants management activities, and how a research effort such as ours can lead to significant reductions in process cycle time and operational cost. However, there remains perhaps an intimately related area for further investigation that we are seek to address. This concerns the activities involved in the establishment, management, and fulfillment of research programs by Federal agencies.

At ONR, the source of problems to be addressed by a research program is often Commands within the USN Fleet. In turn, Commands within the Fleet are also the source of the budget authority providing the funds to be expended in acquiring research results through independent investigations. ONR program officiers (also called science officiers), program managers, and division directors must increasingly organize and manage knowledge brokering processes.

Knowledge brokering refers to the activities of an organizational agency whose brokers (program officiers) find "servers" (researchers) who can "marshall, integrate, and deliver" services and results (create new knowledge, conduct experiments, prototype technology, produce research reports, etc.) for "clients" (the Fleet). The scope of activity that knowledge brokers at ONR must articulate is growing.

Many program officiers must now organize R&D programs that span from basic research studies, through applied research and prototyping studies, to advanced technology demonstrations that address customer needs. These processes presently take years, a decade, or more to complete.

Whether focused only on ONR, or more broadly at any/all Federal reseearch agencies, a number of basic questions can be asked about the processes, architectures, artifacts, support systems, etc, that support Federal knowledge brokering activities: What are these processes, architectures, etc.? How do they work, how do they work best, and how do they go wrong? Can they be computationally modeled, analyzed, simulated, and so forth across their life cycle? How might Federal knowledge brokering process architectures be redesigned for optimal and adaptive performance? Can their cycle time and cost be substantially reduced? Can the quality and other customer satisfaction criteria for knowledge services and results be systematically improved? Can large research programs be made more affordable, timely, and of higher yield through improved understanding of "the science of science research program management"?

We believe questions such as these merit further investigation. Such investigation is likely to be within the self-interests of:

- the Federal and institutional customers who want scientific research to be done,
- the community of Federal research agencies who administer the Nation's annual multi-billion investment in scientific research programs and projects,
- the community of researchers, particularly those most fluent in the relevant disciplines enabling intelligent integration of information and supporting systems,

through such a field of inquiry or program initiative.

We therefore welcome the opportunity to discuss matters such as these, together with our research experiences outline in this paper, at the Workshop on R&D Opportunities for Federal Information Services.

Biographies

- *Walt Scacchi* is Director of the ATRIUM Laboratory at USC and research professor in the IOM department in the Marshall School of Business. He has been on the faculty at USC since 1981, after completing his Ph.D. in computer science at UC Irvine.
- *John Noll* is a research associate at the ATRIUM Laboratory at USC. He completed his Ph.D. in computer science at USC in 1996. He has worked with Dr. Scacchi since 1988, and in the ATRIUM Laboratory since its inception in 1993.
- *Cedric Knight* is President and CEO of New Directions Technologies Inc., a company he founded in 1995. Prior to this, Mr. Knight was a Commander in the Supply Corp, US Navy (retired 1994) and Director of Procurement at the Naval Air Warfare Center--Weapons Division, at China Lake, CA, a position he held from 1988-1994. In addition, he has also directed or commanded other Navy procurement, logistics, transportation, and supply centers during his military career. Mr. Knight serves as a sub-contractor on this project.
- *Captain Jay Miller*, SC, USN, is Director of Acquisition for the Office of Naval Research, a position held since 1994. Prior to this position, Capt. Miller has directed or commanded various Navy

procurement, logistics, and supply centers. Captain Miller retires from the USN on 1 May 1997.

Acknowledgements

Preparation of this report was supported by a grant from the Office of Naval Research (contract number N00014-94-1-0889). No endorsement implied. None of the material in this report should be construed as a statement of policy, procedure, or preference by ONR, the US Navy, or any other US government agency. Contributors in the ONR Acquisition Directorate participating in this study include Jim Carbonara, Frank O'Day, Diana Nichols, John Starcher, and many others. In addition, former ATRIUM Laboratory researchers, Dr. Peiwei Mi, now at Andersen Consulting, and Dr. Mark Nissen, now at Naval Postgraduate School, contributed to the concepts and tools employed in this investigation.

References

BS96 B.W. Boehm and W. Scacchi, Simulation and Modeling for Software Acquisition (SAMSA). Final Report, Center for Software Engineering, USC, Los Angeles, CA, March 1996. http://sunset.usc.edu/SAMSA/samcover.html

KMW96
R.T. Kouzes, J.D. Meyers, and W.A. Wulf. Collaboratories -- Doing Science on the Internet, *Computer*, 29(8):40-48, August, 1996.

MS96
P. Mi and W. Scacchi. A Meta-Model for Formulating Knowledge-Based Models of Software Development. *Decision Support Systems*, 17(3):313-330. 1996. http://www.usc.edu/dept/ATRIUM/Papers/Process_Meta_Model.ps

Ni94 M. Nissen. Valuing IT through Virtual Process Measurement. *Proc. 15th. Intern. Conf. Information Systems*, Vancouver, Canada, 309-323. December 1994. http://www.usc.edu/dept/ATRIUM/Papers/Process_Measurement.ps

Ni96 M. Nissen. *Knowledge-Based Organizational Process Redesign: Using Process Flow Measures to Transform Procurement*, unpublished Ph.D. Dissertation, IOM Dept., USC School of Business Administration, Los Angeles, CA, March 1996.

NS91 J. Noll and W. Scacchi. Integrating Heterogeneous Information Repositories: A Distributed Hypertext Approach, *Computer*, 24(12):38-45, December 1991. http://www.usc.edu/dept/ATRIUM/Papers/Distributed_Hypertext.ps

NS97a
J. Noll and W. Scacchi. Supporting Distributed Configuration Management in Virtual Enterprises. *Proc. 7th. Software Configuration Management Workshop*, R. Conradi (ed.), Boston, MA, Lecture Notes in Computer Science, Volume TBD, Springer-Verlag, New York.(to appear, 1997). http://www.usc.edu/dept/ATRIUM/Papers/DHT-SCM7.ps

NS97b
J. Noll and W. Scacchi. Supporting Software Development Projects in Virtual Enterprises. (submitted for publication, January 1997) http://www.usc.edu/dept/ATRIUM/Papers/DHT-VE97.html

SM96
W. Scacchi and P. Mi. Process Life Cycle Engineering: A Knowledge-Based Approach and Environment. *Intern. J. Intelligent Systems in Accounting, Finance, and Management*, (to appear, 1997). http://www.usc.edu/dept/ATRIUM/Papers/Process_Life_Cycle.html

Appendix: Examples of ONR Grants Management process redesign alternatives

Diagnosis	*Applicable Hueristics*	*Expected ROI*
Manual step sequence	Consolidate and automate	Med-High
Linear step sequences	Identify parallelization opportunities	High
Many reviews steps	Joint collaborative reviews	High
Many data validation steps	Rule-based review system	Med-High
Many data validation steps	Push validation responsiblities upstream	Med-High
Manual assembly of compound documents	Rule-based document builder	Low-Med
Duplicating and circulating documents	Automate distribution and archiving	Med-Very High
Replace paper documents	Employ electronic proposals and grant documents	High-Very High
Islands of automation	Intranet with process support, data integration, and product navigation	Low-High
Wide-area workflow	Internet-based process enactment	Med-High

Process diagnosis results from a set of analysis routines and procedures that we have developed for "measuring" and classifying process flowgraphs or sub-graph patterns [Ni94,Ni96].

Applicable hueristics are selected from a growing base of experience and published studies of successful process redesign tactics.

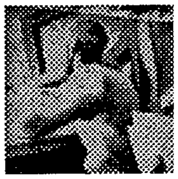
Expected ROI (return on investment) represents the anticipated payoff determined from qualitative or intuitive assessments by ONR personnel and the research team, under the assumption that in-house ONR staff or external contractors could be engaged and funded to implement the necessary redesigns, albeit in a cost-effective and timely manner. Thus, these are simply subjective judgements of the participants. All alternatives, except the rule-based document builder, are presently being investigated or implemented. The outcome of some of these redesigns, such a process step consolidation, automation, and parallelization, have led to, for example, a collapse of 31 process steps in ONR Grant Administration into 1 step, and the compression of 24 steps for Grant Award into 5 steps (or 3 steps, when invoking Federal Reinvention Laboratory waivers on FAR-induced paperwork requirements applicable to ONR). Similarly, the employment of electronic grant proposals elminates a multi-million operational cost associated with the processes that handle conventional paper-based research proposals, but in the context of the other process alternatives.

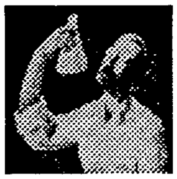
This document was last updated on Wednesday 16 April 1997 at 12:00pm.

PROCESS-DRIVEN INTRANETS: Life-Cycle Support for Process Reengineering

WALT SCACCHI AND JOHN NOLL
University of Southern California

Information systems exist within an organizational web that links people, tools, and products with work processes. PDIs explicitly integrate with this web and offer clear advantages over more traditional intranets in organizational process redesign.





Although business process reengineering has been touted throughout the decade as a way to streamline organizations and improve productivity, three out of four BPR efforts reportedly fail outright, fall short of their performance goals, or are otherwise botched.[1] We see primarily two reasons for this. First, most tools and methods for BPR concentrate on upstream process-design activities like modeling and redesign, almost to the exclusion of mid-stream transitional activities and downstream activities that incorporate the implemented processes into work routines. Second, most efforts do not account for the existing "organizational web"—the social *and* technological infrastructure that spans not only computer-based information systems, but also organizational charts, office and building configurations, telecommunications systems, transportation and distribution channels, resource-to-product processes, and workflows.[2]

Designers of intranet- or Web-based networked information systems are increasingly expected to model the organizational web so that an information system can successfully support it.[3] Unfortunately, understanding how people implement, routinize, and incrementally evolve their work processes continues to take a back seat to simpler concerns such as selecting which browsers, servers, and associated resources to use. Thus, the likelihood remains of low-performing organizational

1089-7801/97/$10.00 ©1997 IEEE

processes and marginally usable information systems.[4]

We believe that corporate intranets may offer a successful way to implement more effective systems. In the approach we describe here, *process-driven intranets* support a computer-based representation of an organizational web.[5] This differs from traditional intranets, which have no explicit representation of the processes they support. By directly supporting the explicit modeling of the organizational processes implemented on a wide-area network, PDIs shift the BPR focus from the traditional upstream activities of process modeling and (re)design to midstream and downstream activities, such as facilitating feedback for continuous process improvement.

Our use of PDIs evolved from the requirements of a recent research case study we conducted for the US Office of Naval Research. The study investigated how to formalize and redesign processes for managing research grants, while also adopting a "corporate-wide" intranet. The intranet, which would be used by ONR headquarters and field offices nationwide, would support internal processes related to the funding and tracking of research grants.

In earlier research, we had established a methodology for the life-cycle engineering of an organizational web.[6] Our challenge became how to best redesign ONR's processes so that a wide-area intranet would be central to the solution. In describing our PDI-centered design approach, we draw examples from this case study. ONR estimates operational savings of $10 to $15 million per year with the proposed redesigned processes as well as reductions in the cycle time for processing research grants by a factor of 10. We believe organizations with similar requirements can achieve similar savings.

CHARACTERISTICS OF A PDI

An intranet is a TCP/IP network that provides a foundation for sharing information in a corporate setting. It supports subsystems for applications such as network management, groupware, and conferencing; and it provides access through a Web browser to corporate information in networked file systems and database management systems. However, business processes that access and update information are buried in programs and dispersed across execution scripts and Web pages. As a result, they can appear disjointed, unstructured, or opaque to many users. Furthermore, changing these business processes often requires programming skill, which disables rather than empowers many process stakeholders. For these reasons, CGI programming languages, document markup languages, or database access languages are poor choices for describing business processes. They do not help people describe, understand, try out, and revise the core processes they routinely perform.

PDIs add subsystems that support configurational (re)design, integration, and continuous improvement of organizational processes. PDIs can be implemented in different ways, for example, through workflow and database management systems, network operating environments, or ad hoc Web-based scripting. In our case, they *require* a formal business-process model that can be represented as a semantic hypertext graph. PDIs can use this representation to interpret a process model so that, for example, links to Web pages or frames are automatically updated where process actions occur. An external link repository and server manages the links. This contrasts with conventional intranets that use handcrafted links and CGI programs to deliver business transaction steps, which require tedious editing and reprogramming to accommodate process changes.

The PDI we developed for ONR was based on concepts, tools, and techniques developed at the University of Southern California's ATRIUM Laboratory to support a multi-year research project investigating ways to reengineer financial operations for acquisition and procurement. Central to this work has been the development of organizational process architectures (OPA) in a variety of operations: large-scale software engineering, new product development, supply chain logistics, corporate finance, and military procurement. OPAs model the processes, products, people (roles and work groups), information infrastructure, and tools central to an organization's routine work operations. They constitute a persistent information resource that can be tailored for reuse throughout an organization or, with a composite OPA, throughout an industry.[7]

Process Modeling Language

We specify an OPA using the PML process modeling language,[8] a notation we developed for defining a minimal set of objects, attributes, and relations common to processes, products, user roles, and tool and application invocations. PML specifies process steps within common control-flow structures, including concurrency. Within each process step it specifies

- server-side data resources or products (input or output),
- client-side tools or forms used, and
- user roles.

It also specifies a role-based user interface. Further, it can specify pre- and post-conditions that act as constraints on input and as goals for output values for objects and products accessed or updated per step.

Figure 1 on the next page is a sample PML description for a proposal submission subprocess within ONR's Preaward process. This process has two top-level steps, submit_proposal and submit_supporting_documents. In the first step, an HTML-based form captures a proposal file from a principal investigator and a proposal.id is assigned. The second step comprises two concurrent branching actions. The first action requires an e-mail address (captured in an earlier subprocess, which is not shown) to automatically invoke a mail tool, which then sends the designated Web page to the e-mail address. No person or agent is needed to perform this action. The second action involves the submission of a budget file, in a manner similar to

```
process proposal_submit {
    action submit_proposal {
            requires { "proposal" }
            tool {"http://gilligan.usc.edu:3280/process/onr/forms/submit-proposal.html"}
            provides { "proposal.id" }
            agent { "PrincipalInvestigator" }
    }
    branch submit_supporting_docs {
            action submit_certs {
                    requires { "email-address-PI-BusOffice" }
                    tool {sendmail: "http://www.onr.navy.mil/sci%%5Ftech/special/onrpr02.html"}
                    provides { "certification-info-sent" }
            }
            action submit_budget {
                    requires { "budget" }
                    tool {"http://gilligan.usc.edu:3280/process/onr/forms/create-budget.html"}
                    provides { "budget.id" }
                    agent { "PrincipalInvestigator" }
            }
    }
}
```

Figure 1. PML excerpt for a subprocess in submitting proposals to the Office of Naval Research.

submitting a proposal, which produces an assigned budget_id.

Although not shown, all required, provided, or agent entities are declared and assigned object types, similar to object-based modeling schema. Process entities may be related (or linked) by different types of dependencies, such as precedence, availability, and decomposition. Similarly, tools are specified via local/remote command scripts, such as URLs and operating system shell scripts. Details about our tool integration and activation scheme are available elsewhere.[5,8]

Compiler

We also developed a PML compiler to build the semantic representation of processes from the OPA into programs and frame-embedded links for user interaction through a browser. The compiler can generate executable programs in Tcl or Java, for example. The programs update frames in the user's browser with links to planned, pending, available, or completed processes' steps. The compiler can alternatively generate Web page content in JavaScript for process prototypes, depending on the code generator configured into it. The generated process execution code serves as input to the runtime environment—a set of services that act as an "operating system" to execute networked processes (workflow automation) and to access and update files or databases. This network operating environment can then be distributed and integrated with compatibly configured intranet/Web servers for intranet or Web-based delivery, access, and navigation-based process enactment.

This ability to generate PDIs directly from PML specifications allows an organization to rapidly implement, distribute, evaluate, and refine incremental process changes. Further, it lets users develop and implement a composite, intralinked OPA that connects processes, products, and people across a network of organizations on the Internet.

PDI-CENTERED REDESIGN

The systematic design of an OPA is a collaborative effort that requires incremental development, iterative user validation and refinement, rapid process visualization and prototyping, and the cooperative redesign of ad hoc process task instances and models. PDIs are an effective implementation mechanism for this work because they let an organization easily manipulate a formal representation of its organizational web.

In this section we review the activities associated with the design or redesign of OPAs: upstream design, midstream implementation, and downstream routinization.[6] PDIs contribute to each set of activities in different ways. Their principal value in the ONR case study was to support process design and implementation, making it easier to involve process stakeholders and get their input.

Upstream Process Design Activities

Traditional BPR tools tend to limit their focus to activities in this phase: process metamodeling, elicitation and modeling, analysis and simulation, and redesign. These activities help in understanding the as-is situation and identifying possible to-be alternatives. In this phase, PDIs enable the corporate-wide sharing of OPA models, as well as the browsing and capture of feedback from analysis, simulation, and redesign activities. This in turn provides a channel for improving how organizational processes are modeled and redesigned. At this stage, complex organizational activities are quickly redesigned but with less regard for how the redesigned alternatives will be implemented or put into routine practice.

Metamodeling. This activity constructs and refines a concept vocabulary and logic—an ontology—for representing process family instances. This, in turn, requires understanding the domain, context, and organizational web for the processes at hand. In the ONR project, we used an ontology derived from similar studies in modeling and formalizing OPAs for related domains.

Elicitation and modeling. This activity focuses on capturing informal descriptions of processes within the organizational web and then converting them to formal process models and OPAs. Since the models and OPAs take the form of directed attributed graphs with typed links and nodes, they can be represented as semantic hypertexts that can be distributed, configured, browsed, navigated, and updated over a PDI or the Internet.[5,8] One of the major tasks in this activity is to understand existing processes before suggesting process alternatives.

For example, in the ONR effort, we involved personnel from all levels of the organization. We needed to educate them about our motives and methods, and they needed to educate us on the generic, circumstantial, and problematic variants of their grants-management processes. We typically performed two or three iterations of process elicitation, analysis, visualization, and refinement, particularly with key process users or subject-matter experts. We did follow-up validations and refinements with personnel not necessarily involved in these iterations. We also established a set of agreements between the research team and ONR personnel:

- The research team would identify more than one opportunity to optimize the redesign of work processes, information flow, and information integration.
- Effort would be directed at improving personnel effectiveness and workflow without increasing anyone's workload.
- The redesign would create no new personnel positions.
- Processes would be developed in three forms: as-is (legacy form), to-be (alternative process architectures), and transition (steps taken in 30-day increments to transform work patterns and processes from as-is to to-be).
- All intermediate and in-progress results would be posted on the USC ATRIUM Web site so that ONR personnel could access this information at their discretion and provide questions or feedback.
- ONR personnel would select the improvement alternatives to be implemented.
- Any improvements had to be self-motivating or otherwise enable local organizational incentives to increase the likelihood of their successful implementation and incorporation into routine.

As a result of these agreements, we captured models of four grants-management processes in the OPA, coded them into PML, and posted the PML and associated process visualizations derived from the PML on an ATRIUM Web server.

Static analysis and simulation. A process model's static properties can be checked for consistency, completeness, internal correctness, and traceability. Processes modeled in a language-based notation such as PML can use a language compiler or interpreter to analyze static properties. In our experience, this analysis is perhaps the best source of high-value, short-term payoffs during process redesign because it clarifies where processes are and are not understood. PDIs can also provide simple mechanisms for producing intranet- or Web-based management reports or presentation materials suitable for routine use.

Simulation symbolically executes process models to determine the path and flow of intermediate state transitions in ways that can be made persistent and thus be replayed, queried, dynamically analyzed, and reconfigured into multiple alternative scenarios. We have used two kinds of process simulations:[6] *Knowledge-based* simulation enables capabilities such as query-driven and reverse simulations; *discrete-event* simulation offers visual simulations of processes, which make it easy to discover dynamic process bottlenecks and optimization opportunities.

Redesign. Process redesign focuses on reorganizing and transforming the structure of relationships within a modeled process and OPA to address weaknesses, such as too many steps, handoffs, or participants. Process redesign starts by applying a set of analysis routines and procedures we have developed for "measuring" and classifying process flow graphs or subgraph patterns. These measures help to surface process inefficiencies. Redesign heuristics, selected from a growing base of experience and published studies of successful tactics, are then applied to reveal efficient, transformed process representations.[10]

Figure 2 on the next page shows the structure of process flow steps for Funding Approval, a subprocess of ONR's research grants Preaward process. The levels of nesting indicate the chain of review and rework loops, which reveals potential weaknesses in the process, such as the apparent lack of parallelism in the process flow. This in turn suggests a process redesign heuristic that might call for joint collaborative reviews. With joint reviews, ONR could quickly separate the funding requests that are ready for approval or denial from those that require further attention or rework.

By using visualization aids like the graph in Figure 2 with their corresponding PML process descriptions, we identified several process inefficiencies for potential redesigns. For example,

- Consolidate and automate the *manual step sequence.*
- Parallelize *linear step sequences.*
- Collapse the *many work review steps* into joint collaborative review sessions.
- Move *many data validation steps* to earlier in the process.
- Use rule-based systems to automatically configure the *manual assembly of compound documents.*
- Eliminate the *extensive use of paper documents,* instead using electronic versions that can be stored in conventional or hypertext databases, accessible over wide-area networks.
- Use the intranet to consolidate *islands of automation,* integrating data transactions and the invocation of software

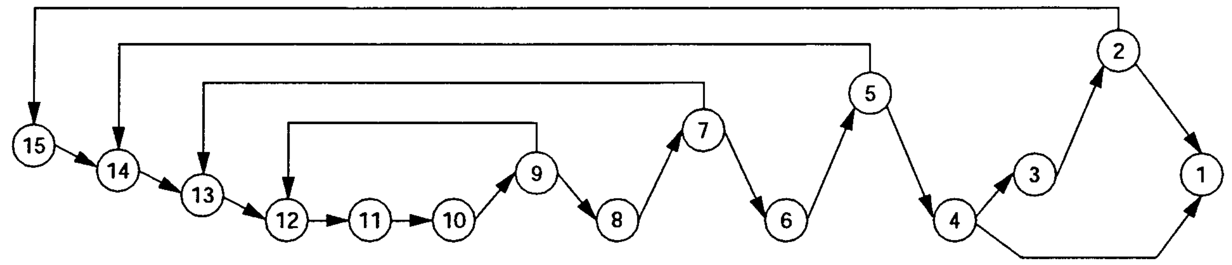


Figure 2. Chain of review approvals in the **Funding Approval** subprocess of ONR's research grants management. Graphs like this make it easier to visualize process weaknesses such as, in this case, a lack of parallelism in reviews.

applications or tools through client-side Web browsers.

- Use intranet-based process enactment mechanisms to better visualize *workflow across multiple organizational units* across the corporate network.

ONR is implementing all these alternatives, except the rule-based document-configuration system. The alternatives have reduced the steps in ONR's Grant Administration process from 31 to one, and for the Grant Award process from 34 to five (and possibly three). Similarly, using electronic grant documents eliminates a multimillion-dollar operational cost associated with the legacy processes that handle conventional paper-based research proposals, when considered with other process redesign alternatives.

Midstream Implementation Activities

Once processes and organizational webs have been redesigned, the next objective is to implement the redesigned alternatives. Planning and executing a transition process is central to the success of implementation. Like the implementation, the transition must engage process stakeholders in a way that lets them share ownership in the process redesign. PDIs help make process redesign alternatives easy to prototype and execute across a corporate network. Participants can thus see and experiment with process alternatives before committing to their final implementation and routinization. The aim of activities during this phase is to support an evolutionary refinement and implementation of the redesigned processes.

Visualization. Process stakeholders and senior managers need to see the structure and flow of process redesign. Graphical views frequently trigger an intuitive understanding of how a process works or doesn't work. Process stakeholders are often surprised by the difference in as-is and to-be process views, which helps to invoke additional feedback on process alternatives.

In Figure 2, for example, the visual display of a process structure helped its users and owners recognize that performing this process without getting caught in a rework loop was highly unlikely. In other words, if the probability of getting through the decision point in each loop is 0.8, with four nested loops, the probability of getting through the decision points of all four loops without performing a rework iteration is approximately 0.4 ($0.8 \times 0.8 \times 0.8 \times 0.8$), meaning most passes through the process will require rework. This process visualization helped the participants quickly recognize the previously unknown pathology of the as-is process structure. It was then easy to motivate process stakeholders to change the process structure.

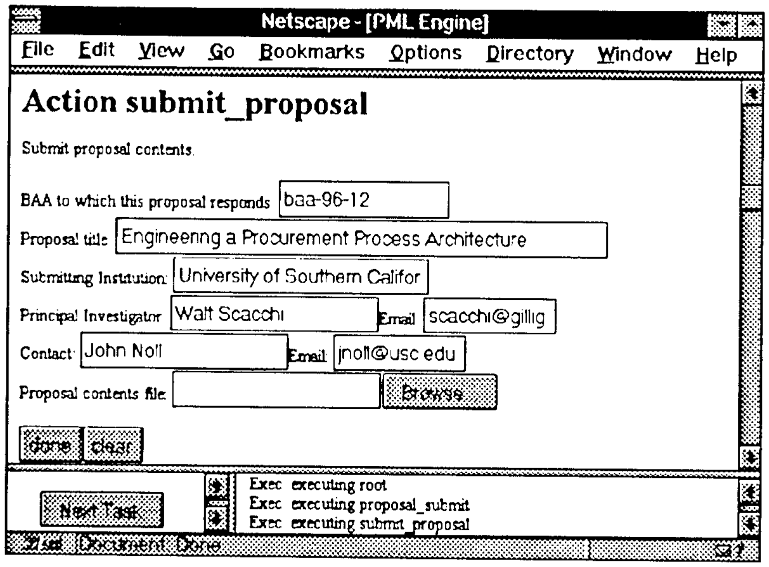
Process visualizations can also be shared across an intranet to make it easier to support feedback on process redesign alternatives among geographically distant stakeholders.

Prototyping. One important way to visualize the dynamics of a process is through prototyping. Prototypes let users preview what will be seen and how it will work when processes are implemented and put into routine operation on an intranet. PDIs let stakeholders access and incrementally traverse prototypes at their desktops through a Web browser, providing a low-cost platform for refining organizational process redesigns.

Figure 3 shows one of many steps in the Grants Management prototype. In this step, which is part of the Preaward process, the principal investigator submits a research project proposal as an HTML document to ONR for consideration. The investigator then enters the data ONR needs to log and track the proposal. This includes identifying the proposal file to be uploaded or transferred to ONR. When the user clicks "done," the transfer is initiated. The user can then select Next Task, through which an electronic budget that accompanies the proposal is submitted. The frame in the bottom right of the figure displays the history of steps completed, serving to record the in-progress status of each step through the process. In this way, multiple concurrent process instances may be active at any time, and users can select the one they want to work on by selecting from available tasks for their selected role and user-ID. Basic access control and authentication mechanisms are applied along the way. Depending on auditing requirements, other attributes, such as date and time of step completion and PC network (IP) address, can be recorded and later replayed or analyzed for process improvement opportunities.[6]

Transition planning and administration. These activities involve assigning and scheduling specified users, tools, and process data objects to a plan for implementing the refined process alternative. PDIs serve as a vehicle for distributing and monitoring progress in fulfilling the plan.

Integration. On the technological side, systems integration entails encapsulating or wrapping selected legacy and new information systems, client-side tools, server-side repositories, and data objects that can be invoked or manipulated when enacting a process instance. Integration provides a computational workspace that binds user, organizational role, task, tools, and input and output resources into "semantic units of work." Scripting languages can be used, as can middleware interface mechanisms, to simplify the integration of distributed objects and services.[8,9]



Action submit_proposal

Submit proposal contents

BAA to which this proposal responds: baa-96-12
Proposal title: Engineenng a Procurement Process Architecture
Submitting Institution: University of Southern Califor
Principal Investigator: Walt Scacchi Email: scacchi@gillig
Contact: John Noll Email: jnoll@usc edu
Proposal contents file:

```
Exec  executing root
Exec  executing proposal_submit
Exec  executing submit_proposal
```

Figure 3. View displaying a process step in a prototype of the **Grants Management** process, which is displayed through a PDI-based browser.

Addressing the organizational integration of processes and systems may pose technical challenges, but, when those challenges are effectively addressed, there is a high payoff. For example, in the ONR case study, we identified a major integration problem in grants management. Many at ONR never knew with certainty the balance and distribution of funds as obligations, invoices, or disbursements. Obligations are budget allocation preferences that could be altered by top-level decision-makers at different times or under different conditions, invoices denote bills submitted for payment (subject to approval), and disbursements represent funds spent. The time between when ONR agreed to fund a project (obligation) and when it transmitted and authorized acceptance of invoices was typically seven to 10 weeks. The time from obligation to disbursement was often many months or longer. During these times, uncertainty about the balance of funds across each category or for different subsets of research grant awards led to conflicts and organizational inefficiency. Reducing these lead times to days (or better yet, hours) would alleviate these problems.

Complicating expenditure management even more was the structure of the information flow. The needed information was distributed across three distinct information systems and across four grants-management processes. Each system had its own administrative authority, organizational location, database, data model, and data format. We proposed a more integrated view, which Table 1 on the next page shows. Each of the last three columns represents data managed by a separate database management system and administrative authority. Integrating these databases into a single global database would be too expensive and time-consuming. A more realistic alternative was to use a form-based tool to provide read-only views to a data cache that periodically queried, downloaded, reformatted, and sorted (using award and request numbers as indices) the funding data from each database.

ONR personnel indicated that having on-demand access to an integrated view of funding actions over an intranet would directly address senior management's critical decision-making needs in ways that had not been available. Stressful ad hoc activities and workarounds for expenditure management could be eliminated. We also found that similar integrated data views could be created to display the dates associated with the performance of specified acquisition process steps, which would meet status reporting needs. Linking these tables with common PC tools like WordPerfect and Microsoft Access would automate and streamline management report preparation even more because the required reports and presentation graphics could then be produced on demand in redesigned grants management processes.

Environment generation. In this activity, a process model is automatically transformed into a PDI that selectively presents prototyped or integrated information systems to end users for process enactment. Our PML compiler and runtime environment make an effective application generator—in this case for generating PDIs that implement partially and fully specified models of complex organizational processes, user roles, tool or application activation, and input/output data mappings.

Downstream Routinization Activities

Once implemented, organizational processes must evolve into the background of the work setting so that competent process users are not constantly reminded what to do, but can rather

Table 1. A proposed integrated view of ONR funds across acquisition processes.

ONR Award Number	Request Number	Obligated Funding	Amount Invoiced	Amount Disbursed
N00014-95-1-0986	96PR03062-00	$100,000.00	$100,000.00	$100,000.00
N00014-94-2-0011	96PR04842-02	$606,317.00	$500,000.00	$350,000.00
N00014-95-2-0014	96PR05343-00	$381,562.00	$81,562.00	$81,562.00
...	...	...	...	...

seek help when something unexpected happens. In routinized processes, process structure and performance details often become tacit knowledge for those who execute them—until something occurs to disrupt the workflow. For core organizational processes, such occurrences are surprisingly frequent. Accordingly, processes must be continually adjusted and refined. PDIs provide an information infrastructure for the activities that support downstream routinization of work.

Instantiation and enactment. In this activity, we perform the modeled process using the PDI. Process enactment guides or enforces users or user roles to apply the process as specified. It looks and operates the same as a process prototype, except that now live applications and data are activated and accessed over the corporate intranet. Our experience is that guidance helps unfamiliar users as well as skilled users facing newly updated or improved processes. Further, PDI workspaces for process enactment are effective in tracking the state of progress made in each concurrent process instance.

Process monitoring, recording, and replay. These activities follow from the ability to collect and measure process enactment data. The history capture frame in Figure 3 is an example of how we might capture performance data for continuous process improvement, particularly when time stamps and other data are also recorded.[9] These activities also aid in documenting what process steps actually occurred in what order. Overall, we have found that such data will be most effective when collected and used primarily by process users, acting individually or in quality circles, rather than as personnel performance measures reported to senior management.

Articulation, evolution, and asset management. These activities shape the ongoing evolution of routinized processes. *Articulation* deals with the diagnosis, repair, and rescheduling of process enactments that have unexpectedly broken down or failed because of some unmet resource requirement.[9] While these problems have been investigated in traditional client-server settings, how to handle them in an intranet or wide-area network environment is still an open problem.

The formalized models and OPA representations for PDIs support process *evolution* for continuous improvement. In the computing world and in complex organizational webs, change is constant. We should expect to incrementally and iteratively enhance, migrate, or reengineer process models, OPAs, and process life-cycle activities to more effectively meet emerging user requirements, and to capitalize on the opportunities new tools and techniques introduce.

Finally, process models and OPAs require *asset management.* This stems in part from the need to systematically support process evolution and continuous improvement. Moreover, in large multisite organizations such as ONR, process assets should be centrally managed, but available on a distributed, wide-area basis. Furthermore, other organizations within an industry or within the federal government may benefit from the reuse, tailoring, or reenactment histories to help move their organizations toward reengineering their in-house processes using PDIs. Thus, PDIs will benefit from the use of database or hypertext repositories for managing and sharing PDI assets.

LESSONS LEARNED

In applying PDIs in the ONR case study, we learned several important lessons. Here we offer a sample of recommendations to those considering experimentation with PDIs.

Take time to understand existing processes. Most existing processes are neither well-defined nor well-understood. Yet, we have found that most process modeling or redesign efforts want to jump to new process alternatives without baselining the existing processes. Process redesign should be structured to encourage sharing and distributing information on proposed process improvements, and PDIs support this.

Make reengineering a collaborative, team-based endeavor. Having intranet tools like PDIs and techniques that support the participation of process stakeholders can help make redesign more successful. Those we developed address how to support process design, implementation, and routinization activities as well as collaboration concerns that other approaches ignore.

Consider multiple redesign alternatives. Some savings aren't possible unless you implement a set of interrelated alternatives.

Recognize that not all processes are suitable for reengineering. PDIs and OPAs may not be suitable for processes that are not large or routinely practiced. Understanding whether and how PDIs and OPAs might be applied to such processes requires additional research.

CONCLUSION

Redesigning an organization's web to more effectively streamline its processes and more efficiently use emerging information technologies is not a simple task. It cannot be finessed by installing a new technology that automates existing processes. A successful redesign requires understanding existing and alternative future forms of how people perform their work processes using available information technology, and engaging these people in transitioning the overall structure of their work.

We believe PDIs represent a new kind of intranet technology that provides a higher level of abstraction and interface for interconnecting distributed organizations through cross-integrated processes. The elements of our approach—PDIs, OPAs, and the design activities described—may form part of the foundation for a new generation of organizations that use process-driven, wide-area networked information systems in their core business processes. ONR is only one of the 10 or so federal agencies in the US government that manage and fund research grants. Collectively these grants total $15 billion annually. Improving and streamlining these processes benefits participants in the academic and industrial research communities. Large corporations also spend millions of dollars on back-office operations associated with funding, procuring, or purchasing goods and services from hundreds of trading partners. Thus, there are a substantial number of organizational settings that are amenable to streamlining and cost savings through reengineering with PDIs.

Our results are based on limited exploration and experimentation. New intranet and Web technologies will also enable others to experiment or apply PDIs, OPAs, and the process life cycle in their organizational web. Process-driven extranets that enable partnering business organizations to interconnect their external commerce processes with their internal operational processes seem to be a ripe area for investigation and commercial exploration. New BPR activities such as process prototyping, process environment generation, and automated process execution using intranets may help in implementing the redesign of business processes as intranet-based work environments. PDIs support incremental, continuous process improvements and enable rapid deployment of these improvements across multiple sites connected through a corporate intranet. ▩

ACKNOWLEDGMENTS

We thank the many people at ONR who contributed to this study, particularly Jim Carbonara, Dennis Padilla, Jay Miller, Diana Nichols, Frank O'Day, Chuck Paoletti, John Starcher, Ann Stuart, Jim Valvocin, and Ralph Wachter. Cedric Knight of New Directions Technologies provided valuable field support and consulting services. Former ATRIUM Laboratory researchers Peiwei Mi, now at Intelligent Systems Technology, and Mark Nissen, now at the Naval Postgraduate School, contributed to the concepts and methods used in this investigation.

This research was supported by ONR grant N00014-94-1-0889. None of the material in this report should be construed as a statement of policy, procedure, or endorsement by ONR, the US Navy, or any other US government agency.

REFERENCES

1. B. Bashein, L.M. Markus, and P. Riley, "Preconditions for BPR Success and How to Prevent Failures," *Information Systems Management*, Vol. 11, No. 2, 1994, pp. 7-13.
2. P. Mi and W. Scacchi, "A Meta-Model for Formulating Knowledge-Based Models of Software Development," *Decision Support Systems*, Vol. 17, No. 3, 1996, pp. 313-330, http://www.usc.edu/dept/ATRIUM/Papers/Process_Meta_Model.ps.
3. J. Tenenbaum, T. Chowdhry, and K. Hughes, "Eco System: An Internet Commerce Architecture," *Computer*, May 1997, pp. 48-55.
4. R. Kling, "Organizational Analysis in Computer Science," *The Information Society*, Vol. 9, No. 2, 1993, pp. 71-87.
5. J. Noll and W. Scacchi, "Supporting Distributed Configuration Management in Virtual Enterprises," in *Software Configuration Management*, R. Conradi, ed., Lecture Notes in Computer Science, Vol. 1235, Springer-Verlag, New York, 1997, pp. 142-160, http://www.usc.edu/dept/ATRIUM/Papers/DHT-SCM7.ps.
6. W. Scacchi and P. Mi, "Process Life Cycle Engineering: A Knowledge-Based Approach and Environment," *Intelligent Systems in Accounting, Finance, and Management*, Vol. 6, 1997, pp.83-107 , http://www.usc.edu/dept/ATRIUM/Papers/Process_Life_Cycle.html.
7. F. Leymann and W. Altenhuber, "Managing Business Processes as an Information Resource," *IBM Systems J.*, Vol. 33, No. 2,:pp. 326-348.
8. J. Noll and W. Scacchi, "Supporting Software Development Projects in Virtual Enterprise," pending publication, original version on http://www.usc.edu/dept/ATRIUM/Papers/DHT-VE97.html.
9. J. Noll and W. Scacchi, "Integrating Heterogeneous Information Repositories: A Distributed Hypertext Approach," *Computer*, Dec. 1991, pp. 38-45, http://www.usc.edu/dept/ATRIUM/Papers/Distributed_Hypertext.ps
10. M. Nissen, "Knowledge-Based Organizational Process Redesign: Using Process Flow Measures to Transform Procurement," PhD dissertation, IOM Dept., USC School of Business Administration, Los Angeles, 1996, http://dubhe.cc.nps.navy.mil/~menissen/dissabst.

Walt Scacchi is a research professor and founding director of the ATRIUM Laboratory at the University of Southern California. He has directed more than a dozen research projects in process engineering over the past 10 years for both corporate and government sponsors. Scacchi received a PhD in information and computer science from the University of California, Irvine. He is a member of the AAAI, ACM, and IEEE Computer Society and an active industry consultant.

John Noll is a software engineer at Network Appliance, Inc., where his interests include wide-area hypertext environments, process enactment mechanisms, and Web caching techniques. During the work reported in this article, he was a research associate at the ATRIUM Laboratory. Noll received an MS and a PhD in computer science from the University of Southern California.

Contact Scacchi at ATRIUM Laboratory,USC, Los Angeles, CA 90089-1421, scacchi@gilligan.usc.edu; or Noll at NetApp, 2770 San Tomas Expwy., Santa Clara, CA 95051, jnoll@netapp.com.